Institut für Architektur von Anwendungssystemen

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelorarbeit

Konzeption und Entwicklung eines Low-Code-Frameworks für Quantencomputing-Anwendungen

Tom Ebert

Studiengang:	Data Science
Prüfer/in:	Prof. Dr. Dr. h. c. Frank Leymann
Betreuer/in:	Daniel Georg, Fabian Bühler
Beginn am:	19. Januar 2024
Beendet am:	30. August 2024

Kurzfassung

Diese Bachelorarbeit widmet sich der Erforschung der Integration von Low-Code-Entwicklungsansätzen in die Quantencomputing-Entwicklung, einem Feld, das aufgrund der hohen Komplexität und der erforderlichen spezialisierten Kenntnisse bislang weitgehend Experten vorbehalten war. Ziel der Arbeit ist es, zu untersuchen, inwieweit etablierte Low-Code-Plattformen, die in der klassischen Softwareentwicklung Anwendung finden, durch gezielte Anpassungen auch für die Entwicklung von Quantencomputing-Anwendungen genutzt werden können.

Die Arbeit stützt sich auf eine systematische Literaturrecherche, die auf den Richtlinien von Kitchenham und Charters basiert. Dabei wurden relevante Publikationen identifiziert und analysiert, um die Möglichkeiten und Grenzen von Low-Code-Werkzeugen im Kontext des Quantencomputings zu beleuchten. Ein besonderes Augenmerk liegt auf Open-Source-Ansätzen und Model-Driven Engineering (MDE), da diese nicht nur die Barriere für den Einstieg in das Quantencomputing senken, sondern auch die Anpassungsfähigkeit und Weiterentwicklung der Plattformen fördern können.

Die Analyse zeigt, dass es zwar bereits erste Ansätze gibt, Low-Code-Plattformen für Quantencomputing nutzbar zu machen, diese jedoch noch in einem sehr frühen Stadium sind. Insbesondere fehlen robuste Open-Source-Tools, die die breite Anwendbarkeit und Skalierbarkeit dieser Technologien gewährleisten. Die Arbeit identifiziert zudem mehrere Forschungslücken, wie etwa die Notwendigkeit, die Effizienz und Anpassungsfähigkeit von Low-Code-Frameworks im Quantencomputing zu verbessern.

Die Ergebnisse dieser Arbeit bieten eine fundierte Grundlage für die Weiterentwicklung von Low-Code-Werkzeugen im Quantencomputing. Zukünftige Forschungen sollten sich darauf konzentrieren, diese Werkzeuge zu optimieren und einen Prototyp zu entwickeln, der die Vorteile von Low-Code und MDE nutzt, um die Komplexität der Quantenprogrammierung weiter zu reduzieren. Damit könnte ein wichtiger Beitrag zur Demokratisierung des Zugangs zu Quantencomputing geleistet werden.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Verzeichnis der Listings

1 Einleitung

Quantencomputing steht an der Schwelle zu einer paradigmatischen Verschiebung in der Informationsverarbeitung, mit dem Potenzial, Problemlösungen zu ermöglichen, die über die Grenzen klassischer Rechenarchitekturen hinausgehen [Sho99a]. Trotz des theoretischen und praktischen Potenzials des Quantencomputings bleibt der Zugang zu dieser Technologie aufgrund der inhärenten Komplexität von Quantenalgorithmen und der erforderlichen tiefgreifenden Kenntnisse in Quantenmechanik und -informatik limitiert [CFK+22]. So unterscheidet sich die Programmierung von Quantencomputern mitunter von der im klassischen Bereich [RP11]. Cerezo et al. [CAB+21] zeigen insbesondere vielversprechende Anwendungen auf, wie die Suche nach Grundzuständen von Molekülen, die Simulation der Dynamik von Quantensystemen und die Lösung linearer Gleichungssystemen.

In der klassischen Softwareentwicklung bieten Low-Code-Entwicklungsumgebungen einen Ansatz, um die Barriere für den Einstieg in die Programmierung zu senken. Dies wird durch die Abstraktion technischer Komplexitäten und die Bereitstellung intuitiver, grafischer Entwicklungswerkzeuge ermöglicht [JMJ+22]. Low-Code-Plattformen bieten auch erfahrenen Entwicklern leistungsfähige Tools. Diese Plattformen ermöglichen durch grafische Layouts eine effizientere Verwaltung komplexer Zustände und verbessern somit die Effizienz und Wartbarkeit des Codes. Die Anwendung von Low-Code-Plattformen erweist sich insbesondere bei einfacheren Projekten als vorteilhaft, stößt jedoch bei komplexeren Anwendungen an ihre Grenzen [BSW22]. In dieser Arbeit werden die Konzepte der Low-Code-Entwicklung und des Model-Driven Engineering (MDE) weitestgehend synonym verwendet. Die tiefergehende Begründung hierfür wird im Abschnitt der theoretischen Grundlagen detaillierter beschrieben. Beide Ansätze zielen darauf ab, die Softwareentwicklung durch den Einsatz von Modellierungstechniken sowohl zu vereinfachen als auch zu beschleunigen und unterstützen sogenannte "Citizen Developer", also nicht-professionelle Entwickler, indem sie es ermöglichen, dass auch diese aktiver in den Entwicklungsprozess eingebunden werden können. Während Model-Driven Engineering (MDE) traditionell auf die formale Modellierung und Transformationen fokussiert ist, liegt der Schwerpunkt der Low-Code-Entwicklung auf der visuellen Programmierung und der Automatisierung vieler Entwicklungsprozesse. Diese Unterschiede sind für die Zielsetzung dieser Arbeit von untergeordneter Bedeutung. Die gemeinsame Verwendung der Begriffe unterstreicht den integrativen Charakter von Low-Code Entwicklung und erleichtert die Diskussion der relevanten Konzepte.

Auch im Bereich der Quantencomputing-Softwareentwicklung wird der Einsatz von Model-Driven Engineering (MDE) bereits erforscht. Gemeinhardt, Garmendia und Wimmer [GGW21a] betonen in ihrer Arbeit die Vorteile von MDE-Techniken, wie domänenspezifischen Modellierungssprachen und generativen Methoden zur Beschleunigung und Vereinfachung der Quantensoftwareentwicklung. Diese Ansätze bieten eine zusätzliche Abstraktionsebene über bestehende Quantenhardware und Programmiersprachen, wodurch eine breitere Nutzung von Quantencomputing durch Fachleute

ermöglicht wird. Diese Perspektive unterstreicht die Relevanz der vorliegenden Arbeit, ein Low-Code-Framework für Quantencomputing-Anwendungen zu entwickeln, das auf MDE-Prinzipien basiert [GGW21a].

In dieser Arbeit wird untersucht, inwiefern Prinzipien der Low-Code-Entwicklung aus dem klassischen Computing übertragbar auf Quantencomputing sind. Basierend auf den Ergebnissen soll die Machbarkeit der Low-Code-Entwicklung für Quantencomputing in zukünftigen Arbeiten untersucht werden können. Bereits vor der Literaturrecherche werden Kriterien festgelegt, die auf grundlegenden Kenntnissen des Quantencomputings basieren. Das Ziel ist es, einen einfacheren Zugang zu Quantencomputern zu ermöglichen, sodass Experten aus verschiedenen Domänen profitieren können, ohne tiefere Fachkenntnisse im Quantencomputing zu benötigen [MR22].

In einem Beitrag von Cabot [CCC20] wird aufgezeigt, dass es im Bereich der Low-Code-Entwicklung derzeit an einer starken Open-Source-Community fehlt. Open-Source-Modellierungswerkzeuge können helfen, bestehende Lücken zu schließen und die damit verbundenen Herausforderungen besser zu adressieren. Während auch Closed-Source-Lösungen dies ermöglichen, bieten Open-Source-Ansätze die zusätzliche Möglichkeit, die Zugänglichkeit und Anpassungsfähigkeit von Low-Code-Plattformen zu verbessern. Dies fördert die Innovationskraft und Zusammenarbeit innerhalb der Community.

Das Hauptziel dieser Arbeit ist die Recherche und Analyse existierender Low-Code-Werkzeuge auf ihre Tauglichkeit als Quantencomputing-Tools. Diese Untersuchung bildet die Grundlage für die Entwicklung eines Frameworks, das die Prinzipien der Low-Code-Programmierung mit den Anforderungen des Quantencomputings verknüpft. Als ergänzendes Werkzeug wird eine prototypische Implementierung einer grafischen Sprache für Quantencomputer erstellt. Zur Erreichung dieses Ziels werden folgende Aufgaben definiert:

- Eine Recherche und Analyse der relevanten Literatur, um ein Verständnis der Low-Code-Konzepte im klassischen Computing zu entwickeln, insbesondere in Bezug auf ihre Möglichkeiten und Grenzen.
- Entwicklung der methodischen Grundlagen und des theoretischen Rahmens für ein minimales Low-Code-Quantencomputing-Framework, basierend auf der Analyse übertragbarer Konzepte klassischer Low-Code-Werkzeuge, die zur Erstellung eines Prototyps verwendet werden können.

Aufbau der Arbeit

Die Arbeit ist in folgender Weise gegliedert:

Kapitel 2 - Theoretischer Hintergund: Hier werden die theoretischen Grundlagen dieser Arbeit beschrieben, vor allem Low-Code-Entwicklung, Model-Driven Engineering und Quantencomputing.

Kapitel 3 - Systematische Literaturrecherche: Beschreibt die Methodik der systematischen Literaturrecherche sowie die Ergebnisse und Interpretation dieser.

Kapitel 4 - Fazit und Ausblick: Hier werden die Ergebnisse der Arbeit zusammengefasst und Anknüpfungspunkte vorgestellt.

2 Theoretischer Hintergrund

Dieser Abschnitt erläutert die grundlegenden Konzepte, die für das Verständnis der Arbeit von zentraler Bedeutung sind: Low-Code-Entwicklung, Model-Driven Engineering (MDE) und Quantencomputing. Diese Konzepte bilden das Fundament für die Entwicklung eines Low-Code-Werkzeugs, das darauf abzielt, die Anwendung von Quantencomputing zugänglicher und benutzerfreundlicher zu gestalten. Im Folgenden werden die genannten Konzepte im Detail dargestellt und ihre Relevanz herausgearbeitet.

2.1 Quantencomputing

Quantencomputing ist ein aufstrebendes Feld der Informatik, das auf den Prinzipien der Quantenmechanik basiert. Im Gegensatz zu klassischen Computern, die auf Bits basieren, nutzen Quantencomputer Qubits. Diese Fähigkeit beruht auf den quantenmechanischen Eigenschaften von Superposition, Verschränkung und Quanteninterferenz, die es Quantencomputern ermöglichen, komplexe Berechnungen parallel und effizienter durchzuführen als klassische Computer. Dieser Abschnitt beleuchtet die Potenziale des Quantencomputing, die spezifischen Einstiegshürden sowie die Herausforderungen, denen sich Entwickler im Vergleich zur klassischen Programmierung gegenübersehen.

Die Quantenmechanik, die das Verhalten subatomarer Teilchen beschreibt, basiert auf einigen grundlegenden Prinzipien, die sie von der klassischen Physik unterscheiden. Eines dieser Prinzipien ist die Superposition, die besagt, dass ein Quantenobjekt, wie ein Qubit, sich gleichzeitig in mehreren klassischen Zuständen befinden kann. Während ein klassisches Bit entweder 0 oder 1 sein kann, kann ein Qubit in einer Überlagerung dieser Zustände sein, mathematisch dargestellt als $\alpha|0\rangle + \beta|1\rangle$, wobei α und β komplexe Zahlen sind, die die Wahrscheinlichkeitsamplituden repräsentieren [NC10].

Ein weiteres zentrales Prinzip der Quantenmechanik ist die Verschränkung, die eine starke Korrelation zwischen den Zuständen von zwei oder mehr Quantenobjekten beschreibt. Wenn Qubits verschränkt sind, wird der Zustand eines Qubits instantan durch den Zustand des anderen beeinflusst, unabhängig von der Entfernung zwischen ihnen. Diese Eigenschaft ermöglicht es, Informationen auf eine Weise zu verarbeiten, die in der klassischen Physik nicht möglich ist [EPR35].

Quanteninterferenz tritt auf, wenn die Wahrscheinlichkeitsamplituden von Quantenzuständen konstruktiv oder destruktiv interferieren. Dies bedeutet, dass sich die Wahrscheinlichkeiten bestimmter Ergebnisse verstärken oder abschwächen können. Dieses Phänomen ermöglicht es Quantenalgorithmen, durch gezielte Interferenzen effizientere Berechnungen durchzuführen als klassische Algorithmen [Fey18].

In der klassischen Physik sind die Zustände von Systemen deterministisch und können mit absoluter Genauigkeit vorhergesagt werden. In der Quantenmechanik sind die Zustände von Systemen probabilistisch, und es kann nur die Wahrscheinlichkeit eines bestimmten Zustands vorhergesagt werden [GS18].

Quantenmechanische Effekte wie Verschränkung führen zu Nicht-Lokalität, wobei die Zustände von verschränkten Teilchen unabhängig von ihrer Entfernung instantan korreliert sind. In der klassischen Physik gibt es keine Entsprechung für diese Art der instantanen Wechselwirkung [ADR82].

Quantenschaltkreise bilden die Grundlage für Quantenalgorithmen und sind ein zentrales Konzept in der Quantencomputing-Entwicklung. Ein Quantenschaltkreis besteht aus einer Sequenz von Quantenlogikgattern, die auf Qubits angewendet werden, um Quantenoperationen durchzuführen. Diese Schaltkreise nutzen die quantenmechanischen Prinzipien der Superposition und Verschränkung, um parallele Berechnungen durchzuführen, die in der klassischen Computerei nicht möglich sind.

Zu den grundlegenden Quantenlogikgattern gehören das Hadamard-Gatter, die Pauli-Gatter und das CNOT-Gatter. Das Hadamard-Gatter (H) transformiert ein Qubit von einem Basiszustand in eine Superposition beider Basiszustände, mathematisch ausgedrückt als $H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ und $H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. Diese Operation ist entscheidend, um die quantenmechanische Eigenschaft der Superposition zu nutzen [GS18].

Die Pauli-Gatter (X, Y und Z) entsprechen den klassischen NOT-, Y- und Z-Operationen und ändern den Zustand eines Qubits entlang der x-, y- oder z-Achse auf der Bloch-Kugel. Zum Beispiel invertiert das X-Gatter (auch als Quanten-NOT-Gatter bekannt) den Zustand eines Qubits, sodass $X|0\rangle = |1\rangle$ und $X|1\rangle = |0\rangle$ [NC10].

Das CNOT-Gatter (kontrolliertes NOT) ist ein Zweiqubit Gatter, das eine Verschränkung zwischen zwei Qubits erzeugt. Es wirkt auf ein Kontrollqubit und ein Zielqubit. Wenn das Kontrollqubit im Zustand $|1\rangle$ ist, invertiert das CNOT-Gatter den Zustand des Zielqubits. Andernfalls bleibt der Zustand des Zielqubits unverändert. Mathematisch wird dies als $CNOT|00\rangle = |00\rangle$, $CNOT|01\rangle = |01\rangle$, $CNOT|10\rangle = |11\rangle$ und $CNOT|11\rangle = |10\rangle$ dargestellt. Dieses Gatter ist essenziell für die Implementierung von Quantenalgorithmen, die Verschränkungen nutzen [Fey18].

Die Zusammensetzung von Quantenoperationen in einem Quantenschaltkreis erfolgt durch die sequentielle Anwendung dieser Gatter auf die Qubits. Die Reihenfolge und Kombination der Gatter bestimmen die durchgeführte Quantenoperation und letztendlich das Ergebnis der Berechnung. Zum Beispiel besteht der Shor-Algorithmus zur Faktorisierung großer Zahlen aus einer komplizierten Sequenz von Quantenoperationen, die Superposition und Verschränkung nutzen, um effizienter zu sein als klassische Algorithmen [Sho99b].

Die Entwicklung und Implementierung von Quantenschaltkreisen stellt jedoch erhebliche Herausforderungen dar. Die präzise Steuerung und Manipulation von Qubits erfordert spezialisierte Kenntnisse die über die klassischen Programmierkenntnisse hinausgehen. Außerdem sind Quantenoperationen anfällig für Fehler und Dekohärenz, was die Zuverlässigkeit und Genauigkeit der Berechnungen beeinträchtigen kann [Pre18].

Zu den bedeutendsten Quantenalgorithmen zählen der Shor-Algorithmus, der Grover-Algorithmus und der Deutsch-Jozsa-Algorithmus. Der Shor-Algorithmus, der zur effizienten Faktorisierung großer Zahlen verwendet wird, hat wesentliche Implikationen für die Kryptografie, da viele Verschlüsse-

lungssysteme auf der Schwierigkeit der Faktorisierung basieren [Sho99b]. Der Grover-Algorithmus bietet eine quadratische Beschleunigung bei der Suche in unsortierten Datenbanken und kann in Anwendungen mit großen Datenmengen von Vorteil sein [Gro96]. Der Deutsch-Jozsa-Algorithmus ermöglicht es, die Konstanz einer Funktion exponentiell schneller zu bestimmen als klassische Methoden, und dient als Demonstration der Leistungsfähigkeit von Quantenalgorithmen [DJ92].

In der Kryptografie ermöglicht die Quantenkryptografie die Entwicklung von Kommunikationsprotokollen, die auf den Prinzipien der Quantenmechanik basieren und gegenüber Abhörversuchen resistent sind [BB14]. Optimierungsprobleme, wie das Travelling Salesman Problem, können durch Quantenalgorithmen effizienter gelöst werden, was in der Logistik und anderen Industriezweigen von Nutzen ist [FGGS00]. Zudem ermöglichen Quantenalgorithmen die Simulation von Quantenmechanischen Systemen, was Fortschritte in den Materialwissenschaften und der Chemie unterstützen kann [ADLH05]. Prognosen zur Entwicklung der Quantencomputing-Technologie deuten darauf hin, dass in den kommenden Jahrzehnten signifikante Fortschritte zu erwarten sind [LJL+10].

2.2 Low-Code-Entwicklung und Model-Driven Engineering

Low-Code-Entwicklung ist ein Ansatz, der darauf abzielt, die Entwicklung von Anwendungen durch die Automatisierung von Prozessen und die Reduzierung der Notwendigkeit von handgeschriebenem Code zu vereinfachen. Dieser Ansatz ermöglicht es Fachexperten, Anwendungen zu erstellen, ohne umfassende Programmierkenntnisse zu besitzen. Low-Code-Plattformen bieten eine Vielzahl von Funktionen, um Anwendungen zu erstellen, zu testen und zu implementieren. Dazu gehören unter anderem visuelle Entwicklungsumgebungen, die es ermöglichen, Anwendungen durch das grafische Anordnen und Modellieren von Komponenten zu erstellen, sowie Datenbanken und anderen Systeme zu integrieren [BBFF21b]. Low-Code-Plattformen bieten auch Funktionen zur Automatisierung von Prozessen, um Anwendungen zu erstellen, die auf Ereignisse reagieren und sich an veränderte Bedingungen anpassen können, wodurch die Erstellung umfangreicher, datengetriebener Anwendungen vereinfacht wird [BBFF21b]. Shridhar [SS21] sowie Alamin und Iqbal [AAM+21] zeigen auf, dass diese Funktionen nicht nur die Effizienz steigern, sondern auch die Skalierbarkeit und Flexibilität der entwickelten Anwendungen erhöhen.

Ein zentrales Merkmal von Low-Code-Plattformen sind visuelle Entwicklungsumgebungen. Diese Umgebungen ermöglichen es, Anwendungen durch grafisches Anordnen und Modellieren zu erstellen [BBFF21b]. Dadurch wird der Entwicklungsprozess nicht nur beschleunigt, sondern auch intuitiver gestaltet, sodass auch Nutzer ohne tiefgehende Programmierkenntnisse in der Lage sind, komplexe Anwendungen zu entwickeln. Hintergrund ist, dass visuelle Darstellungen für Laien einfacher verständlich sind, als Programmiercode. Shridhar [SS21] betont, dass diese visuellen Entwicklungswerkzeuge die Barriere für die Softwareentwicklung erheblich senken und die Produktivität erhöhen, indem sie die Notwendigkeit, komplexe Programmiersprachen zu benutzen, minimieren. Low-Code-Plattformen bieten Funktionen, die es ermöglichen, Anwendungen weitgehend ohne traditionellen, textbasierten Code zu entwickeln. Stattdessen werden Prozesse und Logiken durch visuelle Modelle und Konfigurationen abgebildet [BBFF21b]. Dies reduziert nicht nur die Fehleranfälligkeit, sondern erleichtert auch die Wartung und Anpassung von Anwendungen. Alamin und Iqbal [AAM+21] führen aus, dass diese Ansätze Entwickler dabei unterstützen, sich auf die Geschäftslogik und die spezifischen Anforderungen der Anwendung zu konzentrieren, anstatt sich mit den Feinheiten der Programmierung auseinanderzusetzen.

Ein weiterer Vorteil der Low-Code-Entwicklung ist die Förderung der Zusammenarbeit zwischen verschiedenen Interessensvertretern wie Geschäftsanalysten, Entwickler und Endnutzern. Diese engere Kollaboration führt zu einer besseren Übereinstimmung der entwickelten Software mit den geschäftlichen Anforderungen [SS21]. Alamin und Iqbal [AAM+21] untersuchen in ihrer empirischen Studie die Herausforderungen, denen sich Entwickler in Low-Code-Umgebungen gegenübersehen, und zeigen auf, dass trotz der Vorteile auch Aspekte wie Plattformabhängigkeit und eingeschränkte Anpassungsfähigkeit bedacht werden müssen.

Trotz der zahlreichen Vorteile, die Low-Code-Plattformen bieten, sind sie nicht frei von Herausforderungen und Risiken. Eine wesentliche Sorge ist die Abhängigkeit von der gewählten Plattform. Diese sogenannte Plattformabhängigkeit kann sowohl bei Organisationen als auch bei individuellen Entwicklern dazu führen, dass es schwierig wird, die entwickelten Anwendungen auf andere Systeme zu übertragen, falls dies erforderlich wird. Dies könnte langfristige Abhängigkeiten von einem bestimmten Anbieter zur Folge haben und die Flexibilität bei zukünftigen Entwicklungen einschränken [AAM+21].

Darüber hinaus besteht die Gefahr, dass die Anpassungsmöglichkeiten der von Low-Code-Plattformen generierten Anwendungen begrenzt sind. Insbesondere bei spezifischen, nicht standardisierten Anforderungen stoßen solche Plattformen oft an ihre Grenzen, was zu suboptimalen Lösungen führen kann [AAM+21]. Auch Sicherheitsaspekte können ein Risiko darstellen, da die Verwendung von vorgefertigten Bausteinen potenziell Schwachstellen mit sich bringen kann, die nicht immer leicht zu identifizieren sind. Es ist daher unerlässlich, diese potenziellen Risiken zu berücksichtigen und sorgfältig abzuwägen, ob die Vorteile die Nachteile überwiegen, insbesondere in spezialisierten Anwendungsbereichen wie dem Quantencomputing, wo hohe Flexibilität und maßgeschneiderte Lösungen von besonderer Bedeutung sein können. Die wesentlichen Charakteristika der Low-Code-Entwicklung sowie deren Bewertung als Chance oder Risiko, sind in Tabelle 2.1 zusammengefasst.

Model-Driven Engineering (MDE) ist ein Ansatz zur Softwareentwicklung, bei dem Modelle als zentrale und treibende Elemente des Entwicklungsprozesses fungieren. In MDE dienen diese Modelle nicht nur zur Unterstützung, sondern bilden die Grundlage für die Generierung des Quellcodes. Dieser Ansatz ermöglicht es, durch formale Modelle eine abstrakte und detaillierte Darstellung der Anwendung und ihrer Funktionalität zu schaffen. Diese Modelle werden anschließend zur automatischen Codegenerierung verwendet, was zu einer beschleunigten Entwicklung, einer erhöhten Wiederverwendbarkeit und einer verbesserten Qualität der Anwendungen führt [Sel03].

Ein wesentliches Merkmal von MDE ist die Nutzung formaler Modelle zur Softwareentwicklung. Diese Modelle bieten eine präzise und verständliche Beschreibung der Struktur und des Verhaltens der zu entwickelnden Software [Sch06]. Durch die Abstraktion von technischen Details ermöglichen formale Modelle eine klare Kommunikation zwischen den verschiedenen Interessensvertretern im Entwicklungsprozess und tragen dazu bei, Missverständnisse zu vermeiden. Formale Modelle können in verschiedenen Formen vorliegen, darunter UML-Diagramme, ER-Diagramme und BPMN-Modelle, die jeweils spezifische Aspekte der Software und ihrer Prozesse abbilden [Sel03].

Ein weiteres zentrales Element von MDE ist die Transformation von Modellen in Code. Dieser Prozess, der als Model-to-Code-Transformation bezeichnet wird, ermöglicht es, aus den erstellten Modellen automatisch lauffähigen Code zu generieren [BCW17]. Dies führt nicht nur zu einer Beschleunigung der Entwicklung, sondern reduziert auch die Fehleranfälligkeit, da die Notwendigkeit von textbasiertem Code minimiert wird. Durch die Automatisierung der Codegenerierung wird

Charakteristik	Beschreibung	Chance/Risiko
Visuelle Entwicklungsumgebungen	Anwendungen werden durch grafisches Anordnen und Modellieren von Komponenten erstellt.	Chance: Erhöht die Zugänglichkeit und Benutzerfreundlichkeit
Minimierung von textbasiertem Code	Prozesse und Logiken werden durch visuelle Modelle und Konfigurationen abgebildet, ohne traditionellen Code.	Chance: Reduziert Fehleranfälligkeit, Risiko: Eingeschränkte Flexibilität
Automatisierung von Prozessen	Anwendungen reagieren auf Ereignisse und passen sich an veränderte Bedingungen an.	Chance: Steigert Effizienz und Anpassungsfähigkeit
Integration von Datenbanken und Systemen	Einfachere Erstellung datengetriebener Anwendungen durch Integration mit Datenbanken und anderen Systemen.	Chance: Erhöht die Funktionalität und Datenzugänglichkeit
Engere Kollaboration verschiedener Interessensvertreter	Erleichtert die Zusammenarbeit zwischen Geschäftsanalysten, Entwicklern und Endnutzern.	Chance: Verbessert die Übereinstimmung der Software mit den Geschäftsanforderungen
Schnellere Entwicklungszeiten	Verkürzt die Entwicklungszeiten durch reduzierte Programmieranforderungen.	Chance: Erhöht die Produktivität
Zugänglichkeit für Nicht-Programmierer	Ermöglicht auch Fachexperten ohne tiefgehende Programmierkenntnisse die Entwicklung von Anwendungen.	Chance: Erweitert den Entwicklerkreis
Flexibilität und Skalierbarkeit	Erhöht die Flexibilität und Skalierbarkeit der entwickelten Anwendungen.	Chance: Erleichtert Anpassungen und Wachstum
Plattformabhängigkeit	Abhängigkeit von spezifischen Low-Code-Plattformen und deren Limitierungen.	Risiko: Erhöht die Abhängigkeit und mögliche Einschränkungen
Wartung und Anpassung	Erleichtert die Wartung und Anpassung von Anwendungen durch visuelle Modelle.	Chance: Verbessert die Wartbarkeit und Anpassungsfähigkeit

Tabelle 2.1: Charakteristika der Low-Code-Entwicklung und deren Bewertung als Chance oder Risiko [AAM+21; BBFF21b; SS21]

zudem die Konsistenz zwischen den Modellen und dem finalen Code sichergestellt. Verschiedene Werkzeuge und Plattformen wie das Eclipse Modeling Framework (EMF) [Webb] und Acceleo [Weba] unterstützen diesen Transformationsprozess und ermöglichen eine nahtlose Integration in den Entwicklungsworkflow [SBMP08].

MDE fördert außerdem die Wiederverwendbarkeit von Modellen und Komponenten. Durch die Definition wiederverwendbarer Modellbausteine können Entwickler effizienter arbeiten und die Qualität der Anwendungen erhöhen. Dies ermöglicht es, bewährte Lösungen und Muster zu nutzen und so den Entwicklungsaufwand zu reduzieren [FR07a]. Zusätzlich trägt MDE zur Verbesserung der Wartbarkeit und Erweiterbarkeit von Software bei, da Änderungen am Modell automatisch im generierten Code reflektiert werden können.

Trotz der Vorteile gibt es auch Risiken und Herausforderungen, die bei der Anwendung von MDE beachtet werden müssen. Zu den Risiken gehört die potenzielle Komplexität der Modelle, die schwer zu verstehen und zu pflegen sein können. Darüber hinaus besteht eine Abhängigkeit von speziellen Modellierungs- und Code-Generierungswerkzeugen, was die Flexibilität einschränken kann. Die Qualität des automatisch generierten Codes kann variieren und zu Performance-Problemen und Fehlern führen. Zudem erfordert MDE spezifische Schulungen und Wissen, was zusätzliche Kosten und Aufwand bedeutet. Herausforderungen bestehen auch in der Skalierbarkeit und Integration von MDE-Lösungen in bestehende Systeme sowie in der Modell-zu-Code-Abstraktionslücke, die dazu führen kann, dass nicht alle Details und Optimierungen im generierten Code abgebildet werden [FR07b].

Zusammenfassend lässt sich sagen, dass MDE eine umfassende Methodik zur Softwareentwicklung bietet, die durch die Nutzung formaler Modelle und die Automatisierung der Codegenerierung sowohl die Effizienz als auch die Qualität der entwickelten Anwendungen steigern kann. Die wesentlichen Aspekte und Vorteile von MDE sind in Tabelle 2.2 zusammengefasst. Diese Übersicht bietet eine klare Struktur zur Einordnung der MDE-Methoden, indem sie die verschiedenen Merkmale, deren Beschreibungen und die daraus resultierenden Vorteile und Risiken aufzeigt.

Unterschiede zwischen Low-Code und MDE

Low-Code-Entwicklung und Model-Driven Engineering (MDE) sind zwei Ansätze zur Vereinfachung und Beschleunigung der Softwareentwicklung. Obwohl sie ähnliche Ziele verfolgen, unterscheiden sie sich in ihrer Fokussierung, Zielgruppe und Methodik. Die Low-Code-Entwicklung richtet sich primär an sogenannte „Citizen Developers“, also Anwender mit wenig oder keiner Programmiererfahrung, die dennoch in der Lage sein wollen oder können, Anwendungen zu entwickeln. Dieser Ansatz zeichnet sich durch eine benutzerfreundliche Gestaltung aus, die eine intuitive und visuelle Programmierung ermöglicht. Durch grafische Benutzeroberflächen, bei denen Elemente per Maus verschoben und angeordnet werden können, wird der Entwicklungsprozess stark vereinfacht, was es Anwendern ohne tiefgreifende Programmierkenntnisse erleichtert, aktiv zur Softwareentwicklung beizutragen. Die Hauptstärke der Low-Code-Entwicklung liegt in der schnellen und kosteneffizienten Erstellung von Anwendungen, die auf spezifische Geschäftsanforderungen zugeschnitten sind. Diese Plattformen sind oft cloud-basiert und bieten integrierte Entwicklungsumgebungen, die den gesamten Softwarelebenszyklus unterstützen, von der Anforderungserhebung über das Design bis hin zur Implementierung und Wartung [CCC20].

Im Gegensatz dazu konzentriert sich MDE auf die formale Modellierung und Transformationen und richtet sich vor allem an professionelle Entwickler und Ingenieure. MDE nutzt formale Modelle, um die Struktur und das Verhalten von Software präzise zu definieren. Diese Modelle dienen dann als Grundlage für die automatische Codegenerierung, was besonders in großen und komplexen Projekten von Vorteil ist, bei denen eine hohe Präzision und Konsistenz entscheidend sind. Die Methodik erlaubt es, detaillierte Modelle zu erstellen, die direkt in funktionsfähigen Code umgesetzt werden können. Werkzeuge wie das Eclipse Modeling Framework (EMF) und Acceleo unterstützen diesen Prozess, indem sie die Modellierung und die Transformation von Modellen in ausführbaren Code ermöglichen [DKL+22].

Charakteristik	Beschreibung	Chance/Risiko
Nutzung formaler Modelle	Verwendung von präzisen und verständlichen Modellen zur Beschreibung der Struktur und des Verhaltens der Software.	Chance: Erhöht die Klarheit und Kommunikation zwischen Stakeholdern
Transformation von Modellen in Code	Automatische Generierung von lauffähigem Code aus den erstellten Modellen.	Chance: Beschleunigt die Entwicklung und reduziert Fehleranfälligkeit
Wiederverwendbarkeit von Modellen	Definition wiederverwendbarer Modellbausteine, um Effizienz und Qualität zu steigern.	Chance: Reduziert Entwicklungsaufwand und erhöht Qualität
Verbesserung der Wartbarkeit	Änderungen am Modell werden automatisch im generierten Code reflektiert.	Chance: Erhöht Wartbarkeit und Erweiterbarkeit
Komplexität der Modelle	Erstellung sehr komplexer Modelle, die schwer zu verstehen und zu pflegen sind.	Risiko: Erhöht die Komplexität und Wartungsaufwand
Werkzeugabhängigkeit	Abhängigkeit von speziellen Modellierungs- und Code-Generierungswerkzeugen.	Risiko: Einschränkung der Flexibilität durch Abhängigkeit von Anbietern
Qualität der generierten Codebasis	Variierende Qualität des automatisch generierten Codes.	Risiko: Kann zu Performance-Problemen und Fehlern führen
Kosten und Aufwand für Schulungen	Notwendigkeit spezifischer Schulungen und Wissen für die Nutzung von MDE.	Risiko: Erhöht die Kosten und den Aufwand für Schulungen
Skalierbarkeit und Integration	Herausforderungen bei der Integration und Skalierung von MDE-Lösungen.	Risiko: Komplexität bei groß angelegten Projekten
Modell-zu-Code-Abstraktionslücke	Gefahr, dass Modelle nicht alle Details und Nuancen des Codes abbilden.	Risiko: Fehlen von Details und Optimierungen im generierten Code

Tabelle 2.2: Charakteristika des Model-Driven Engineering (MDE), deren Chancen und Risiken [BCW17; FR07a; SBMP08; Sch06; Sel03]

In dieser Arbeit werden die Unterschiede zwischen Low-Code-Entwicklung und MDE kurz angesprochen, jedoch steht ihre Unterscheidung nicht im Vordergrund der Untersuchung. Der Fokus liegt vielmehr auf den Gemeinsamkeiten beider Ansätze und deren Nutzen im spezifischen Kontext der Arbeit. Während die Low-Code-Entwicklung vor allem darauf abzielt, auch Nutzer ohne tiefgehende Programmierkenntnisse durch vereinfachte visuelle Programmierung in die Lage zu versetzen, Anwendungen zu erstellen, verfolgt MDE das Ziel, die Softwareentwicklung durch formale Modellierung und automatische Codegenerierung zu optimieren und zu beschleunigen. Beide Ansätze teilen das übergeordnete Ziel, die Effizienz der Softwareentwicklung zu steigern und die Komplexität zu reduzieren.

Diese Gemeinsamkeiten sind von zentraler Bedeutung für die Entwicklung einer Low-Code-Plattform im Bereich Quantencomputing. Durch die Reduktion der Programmierkomplexität und die Verbesserung der Zugänglichkeit könnten die Prinzipien beider Ansätze dazu beitragen, die Nutzung von Quantencomputing auf eine breitere Anwenderbasis auszudehnen. Dadurch wird eine Lösung angestrebt, die die Stärken der Low-Code-Entwicklung in Bezug auf Benutzerfreundlichkeit mit den Vorteilen von MDE in puncto Präzision und Automatisierung kombiniert.

Für die Ziele dieser Arbeit ist es weniger entscheidend, ob die Methodik primär auf „Citizen Developers“ oder professionelle Entwickler abzielt. Wesentlich ist vielmehr, dass durch Abstraktion und Automatisierung die Entwicklung von Quantenanwendungen sowohl erleichtert als auch beschleunigt wird. Die Synergie beider Ansätze könnte dazu beitragen, eine breitere Nutzerbasis anzusprechen und gleichzeitig die Entwicklungszeit signifikant zu verkürzen. Darüber hinaus bieten Low-Code-Entwicklung und MDE Potenzial zur Kosteneinsparung, nicht nur in Bezug auf die Entwicklungszeit, sondern auch hinsichtlich der Ressourcen, die für die Implementierung und Wartung der Anwendungen erforderlich sind. In dieser Arbeit wird daher der Fokus darauf gelegt, wie die gemeinsamen Stärken von Low-Code und MDE strategisch eingesetzt werden können, um effiziente und anpassungsfähige Lösungen im Bereich des Quantencomputing zu entwickeln. Die spezifischen Unterschiede zwischen den Ansätzen treten dabei in den Hintergrund zugunsten der Untersuchung ihrer komplementären Eigenschaften.

Gemeinsame Prinzipien und synonymer Gebrauch der Begriffe

Low-Code-Entwicklung und Model-Driven Engineering MDE weisen trotz ihrer Unterschiede in Zielgruppe und Methodik zahlreiche gemeinsame Prinzipien auf. Diese gemeinsamen Prinzipien bilden die Grundlage für die synonyme Verwendung der Begriffe in dieser Arbeit.

Ein zentrales gemeinsames Prinzip ist die Abstraktion. Beide Ansätze zielen darauf ab, die Komplexität der Softwareentwicklung durch die Einführung höherer Abstraktionsebenen zu reduzieren. In der Low-Code-Entwicklung wird dies durch visuelle Programmierung und benutzerfreundliche Oberflächen erreicht, die es auch Nicht-Programmierern ermöglichen, Anwendungen zu erstellen. Im MDE wird Abstraktion durch die Verwendung formaler Modelle erreicht, die eine präzise und verständliche Beschreibung der Softwarestruktur und -funktionalität bieten. Diese Abstraktion erleichtert die Kommunikation zwischen verschiedenen Entwicklern eines Projektes und minimiert die Wahrscheinlichkeit von Missverständnissen und Fehlern.

Ein weiteres gemeinsames Prinzip beider Ansätze ist die Automatisierung. Sowohl die Low-Code-Entwicklung als auch MDE setzen auf automatisierte Entwicklungsprozesse, um die Effizienz und Produktivität zu steigern. In der Low-Code-Entwicklung wird die Automatisierung durch visuelle Entwicklungsumgebungen und grafische Werkzeuge erreicht, die es ermöglichen, ohne umfangreiche Programmierkenntnisse Anwendungen zu erstellen. Diese Werkzeuge erlauben es den Nutzern, Softwareelemente mittels einfacher Handhabung und grafischer Anordnung zusammenzustellen, wobei diese grafischen Modelle dann automatisch in ausführbaren Code umgewandelt werden. Im Gegensatz dazu konzentriert sich MDE auf die Verwendung spezifischer Modelle, die ebenfalls in ausführbaren Code transformiert werden können. Dabei ist zu beachten, dass nicht alle MDE-Modelle für die Codegenerierung geeignet sind. Einige Modelle dienen ausschließlich der Analyse

oder Spezifikation und sind nicht direkt in Code umwandelbar. Die Gemeinsamkeit beider Ansätze liegt hier in der Fähigkeit, modellbasierte Spezifikationen in Code zu überführen, wodurch die Entwicklungszeit verkürzt und die Konsistenz zwischen Modell und Code gewährleistet wird.

Beide Ansätze tragen durch die Einführung von Abstraktionsebenen und die Automatisierung von Entwicklungsprozessen zur Reduzierung der Komplexität in der Softwareentwicklung bei. Dies führt zu einer besseren Verständlichkeit und Wartbarkeit der entwickelten Anwendungen, was wiederum die Qualität der Software verbessert.

In dieser Arbeit werden die Begriffe Low-Code-Entwicklung und Model-Driven Engineering ab hier synonym verwendet. Diese synonyme Verwendung wird durch die gemeinsamen Ziele und Methoden beider Ansätze begründet. Die gemeinsame Betrachtung von Low-Code-Entwicklung und MDE ermöglicht es, Lösungen zu entwickeln, die die Vorteile beider Ansätze kombinieren. Dies kann dazu beitragen, die Hürden für die Nutzung von Quantencomputing zu senken und die Entwicklung von Anwendungen in diesem Bereich zu beschleunigen. Durch die Nutzung der gemeinsamen Prinzipien von Abstraktion und Automatisierung sowie der Reduzierung der Komplexität kann die Softwareentwicklung effizienter und produktiver gestaltet werden.

Zusammenfassend ist in der Tabelle 2.3 eine Übersicht über die gemeinsamen Charakteristika, die in der vorliegenden Arbeit herangezogen werden, um die Vorteile und Potenziale von Low-Code-Entwicklung für die Entwicklung von Quantencomputing-Anwendungen zu verdeutlichen. Diese Charakteristika bilden die Grundlage für die weitere Untersuchung der Anwendung von Low-Code-Entwicklung im Bereich des Quantencomputing.

Charakteristik	**Beschreibung**
Abstraktion	Einführung höherer Abstraktionsebenen zur Reduzierung der Komplexität
Automatisierung	Nutzung von Automatisierung zur Steigerung der Effizienz und Produktivität
Reduzierung der Komplexität	Vereinfachung der Softwareentwicklung durch visuelle Programmierung und formale Modellierung
Steigerung der Effizienz und Produktivität	Verbesserung der Entwicklungsprozesse durch Minimierung manueller Codierung und Nutzung von Modellen

Tabelle 2.3: In dieser Arbeit zugrunde liegende gemeinsame Charakteristika von Low-Code-Entwicklung und MDE

Open Source

Ein weiterer wichtiger Aspekt in der Betrachtung von Low-Code und MDE ist der Einsatz von Open Source Lösungen. Open-Source-Software bietet zahlreiche Vorteile, darunter die Möglichkeit zur gemeinschaftlichen Entwicklung und kontinuierlichen Verbesserung durch eine breite Entwicklergemeinde. Dieser kollaborative Ansatz fördert eine schnelle Identifizierung und Behebung von Fehlern. Zudem sind Open Source Lösungen in der Regel kostengünstiger, da sie keine Lizenzgebühren erfordern, was insbesondere für Forschungsprojekte und kleinere Unternehmen von Vorteil ist [Ray10]. Open Source Plattformen wie GitHub und GitLab bieten umfangreiche Werkzeuge und Ressourcen, die die Entwicklung und Verbreitung von Softwareprojekten erleichtern [Fit06].

Darüber hinaus ermöglichen Open Source Lösungen eine hohe Transparenz und Sicherheit. Der offene Zugang zum Quellcode erlaubt es den Nutzern, den Code zu überprüfen, anzupassen und an spezifische Bedürfnisse anzupassen. Dies erhöht nicht nur die Sicherheit durch unabhängige Überprüfungen, sondern bietet auch die Flexibilität, individuelle Anpassungen vorzunehmen, die in proprietären Systemen oft nicht möglich sind [VV06].

Allerdings gibt es auch Limitationen. Open-Source-Projekte können unter mangelnder Unterstützung und Dokumentation leiden, was die Lernkurve für neue Nutzer steiler macht. Zudem ist die Integration von Open Source Komponenten in bestehende Systeme oft komplex und erfordert spezifisches technisches Know-how [Che06]. Während die breite Verfügbarkeit von Open-Source-Projekten eine große Auswahl bietet, kann dies auch zu einer Fragmentierung führen, bei der es schwierig ist, eine konsistente und stabile Lösung zu finden und zu implementieren.

Trotz dieser Herausforderungen bieten Open Source Lösungen eine flexible und zugängliche Plattform für die Entwicklung und Implementierung von Low-Code und MDE Ansätzen im Quantencomputing, die durch die gemeinsame Nutzung von Wissen und Ressourcen gestärkt wird. Dies ist besonders relevant in einem sich schnell entwickelnden und komplexen Feld wie dem Quantencomputing, wo der Zugang zu neuesten Entwicklungen und die Fähigkeit zur schnellen Anpassung entscheidend sind. Die Nutzung von Open Source Lösungen kann daher nicht nur die Entwicklung beschleunigen, sondern auch die Zusammenarbeit und den Wissensaustausch zwischen verschiedenen Disziplinen und Institutionen fördern [WG06].

2.3 Low-Code-Entwicklung und Model-Driven Engineering im Kontext des Quantencomputing

Im Kontext des Quantencomputing bedeutet dies für die Entwicklung von Low-Code-Werkzeugen, dass diese eine benutzerfreundliche Oberfläche bieten müssen, die es Entwicklern ermöglicht, komplexe Quantenschaltungen zu erstellen, ohne tiefgehende Kenntnisse der Quantenmechanik zu benötigen. Diese Tools sollten Mechanismen zur Fehlerkorrektur und zur Optimierung der Quantenschaltungen beinhalten, um die Zuverlässigkeit und Effizienz der Quantenoperationen zu gewährleisten.

Die Einführung von Low-Code im Quantencomputing könnte potenziell mehrere bedeutende Vorteile bieten. Ein wesentlicher Vorteil ist die Erleichterung der Entwicklung von Quantenalgorithmen. Durch die Bereitstellung von benutzerfreundlichen, visuellen Entwicklungsumgebungen, die die komplexen quantenmechanischen Prinzipien abstrahieren, können Low-Code-Werkzeuge es Entwicklern ermöglichen, Quantenalgorithmen effizienter zu erstellen und zu testen. Dies könnte insbesondere für Entwickler nützlich ein, die über begrenzte Kenntnisse in der Quantenmechanik verfügen, da sie sich auf die Logik und Funktionalität ihrer Algorithmen konzentrieren können, ohne tief in die physikalischen Details eintauchen zu müssen [CCC20].

Ein weiterer Vorteil ist die Verbesserung der Zugänglichkeit für Citizen Developer. Low-Code-Werkzeuge könnten die Barrieren senken. Durch intuitive Schnittstellen und vorgefertigte Module könnten auch Nutzer aus anderen Disziplinen, wie Wirtschaft, Biologie oder Chemie, Quantenalgorithmen entwickeln und anwenden, um spezifische Probleme in ihren Bereichen zu lösen [DKL+22]. Dies könnte zu einer breiteren Akzeptanz und Nutzung von Quantencomputing führen und neue Anwendungen ermöglichen.

Dennoch gibt es Herausforderungen und Limitationen bei der Nutzung von Low-Code-Entwicklung und MDE im Quantencomputing. Eine der größten technischen Herausforderungen besteht in der Integration dieser Tools in bestehende Quantencomputing-Plattformen. Die Komplexität der Quantenhardware und -software erfordert spezialisierte Schnittstellen und Protokolle, um eine nahtlose Integration zu gewährleisten [MK13].

Mögliche Limitationen der Low-Code-Ansätze umfassen begrenzte Anpassungsmöglichkeiten und die Abhängigkeit von spezifischen Plattformen. Low-Code-Werkzeuge bieten in der Regel vordefinierte Module und Bausteine, die zwar die Entwicklung erleichtern, aber möglicherweise nicht die Flexibilität bieten, die für spezialisierte oder fortgeschrittene Quantenanwendungen erforderlich ist. Zudem könnten diese Tools von den spezifischen Technologien und Plattformen abhängen, auf denen sie entwickelt wurden, was ihre Portabilität und Interoperabilität einschränken könnte [NC10].

Die Tabelle 2.4 fasst die potenziellen Vorteile der Low-Code-Entwicklung im Quantencomputing sowie die damit verbundenen Herausforderungen und Hindernisse zusammen, um einen umfassenden Überblick über die Chancen und die technischen Hürden zu bieten, die bei der Implementierung solcher Ansätze berücksichtigt werden müssen.

Potenziale von Low-Code im Quantencomputing	**Herausforderungen und Hindernisse**
Erleichterung der Entwicklung von Quantenalgorithmen durch benutzerfreundliche, visuelle Entwicklungsumgebungen	Integration in bestehende Quantencomputing-Plattformen
Verbesserung der Zugänglichkeit für nichttechnische Nutzer durch intuitive Schnittstellen und vorgefertigte Module	Skalierbarkeit und Performance bei großen Quantenoperationen
Förderung von Kollaboration durch interdisziplinäre Teams und breitere Nutzung	Begrenzte Anpassungsmöglichkeiten bei spezialisierten oder fortgeschrittenen Anwendungen
Beschleunigung der Entwicklungsprozesse durch Automatisierung und Abstraktion der komplexen quantenmechanischen Prinzipien	Abhängigkeit von spezifischen Plattformen und Technologien
Reduzierung der Einstiegshürden für neue Entwickler im Bereich Quantencomputing	Notwendigkeit von Fehlerkorrektur und Management von Dekohärenz und Fehlerraten

Tabelle 2.4: Potenziale und Herausforderungen von Low-Code im Quantencomputing

3 Systematische Literaturrecherche

Die systematische Literaturrecherche (SLR) ist eine methodische Vorgehensweise, um Forschungsfragen systematisch und umfassend zu beantworten. Sie folgt einem klar definierten Prozess, der darauf abzielt, relevante Studien zu identifizieren, zu bewerten und zu synthetisieren [KC+07]. Der Prozess beginnt mit der Planung und Formulierung eines Forschungsprotokolls, das die Schritte und Kriterien für die Literaturrecherche festlegt.

Wesentliche Bestandteile einer SLR sind die strukturierten Schritte zur Suche und Auswahl der Studien, die Datenerhebung und die anschließende Analyse. In der Such- und Auswahlphase werden Studien identifiziert, die die gestellten Forschungsfragen adressieren. Ein festgelegtes Protokoll definiert die Kriterien, die zur Bewertung der Relevanz und Qualität der Studien herangezogen werden. Dieser methodische Ansatz stellt sicher, dass die Ergebnisse transparent, nachvollziehbar und wiederholbar sind, wodurch Verzerrungen minimiert werden [OS15].

Die Datensynthese kombiniert die extrahierten Informationen, um ein umfassendes Bild des Forschungsstandes zu zeichnen. Dies ermöglicht es, allgemeine Trends zu erkennen, bestehende Konzepte zu integrieren und Forschungslücken zu identifizieren. Dank dieser systematischen Vorgehensweise können zuverlässige und fundierte Schlussfolgerungen gezogen werden, die einen erheblichen Mehrwert für die Forschungsgemeinschaft bieten [PFMM08].

3.1 Methodik

Basierend auf den Richtlinien von Kitchenham und Charters [KC+07] wird ein systematisches Literaturreview (SLR) durchgeführt, um ein tiefgehendes Verständnis der gemeinsamen Charakteristika, Möglichkeiten und Hindernisse von Low-Code-Werkzeugen zu erlangen.

Vor der protokollbasierten Suche bietet sich eine explorative Suche an. Diese dient dazu, einen ersten Überblick über die verfügbare Literatur zu gewinnen und die Relevanz der Forschungsfrage zu überprüfen. Dabei werden die Suchbegriffe in den ausgewählten Datenbanken eingegeben und potenziell relevante Ergebnisse festgehalten. Dieser Schritt ermöglicht es, die Breite und Tiefe des Forschungsfeldes zu erfassen und die Suche gegebenenfalls zu verfeinern. Die Ergebnisse der explorativen Suche dienen auch als Ausgangspunkt für die systematische Literaturrecherche.

Zur Visualisierung und Strukturierung des Prozesses dient eine Darstellung, die sich an den Ablauf der systematischen Literaturrecherche von Kitchenham orientiert, jedoch um spezifische Aspekte aus der Arbeit von Brereton et al. [BKB+07] erweitert wurde. Diese Erweiterungen berücksichtigen insbesondere die praktischen Erfahrungen und spezifischen Herausforderungen bei der Durchführung von SLRs im Bereich der Softwareentwicklung. Abbildung 3.1 zeigt diesen angepassten Prozessablauf und verdeutlicht die methodischen Schritte, die zur systematischen und strukturierten Erfassung der relevanten Literatur führen. Diese Schritte bilden die Grundlage für

die nachfolgende Datenanalyse und Interpretation. Die Schritte aus Phase 1, die Forschungsfrage sowie die Methodik, sind in den folgenden Absätzen dokumentiert. Phase 2, die Durchführung der Literaturrecherche, findet sich in Kapitel 3.2. Die Ergebnisse und deren Interpretation als Artefakte der Phase 3 sind in den Kapitel 3.3 und 3.4 dokumentiert.

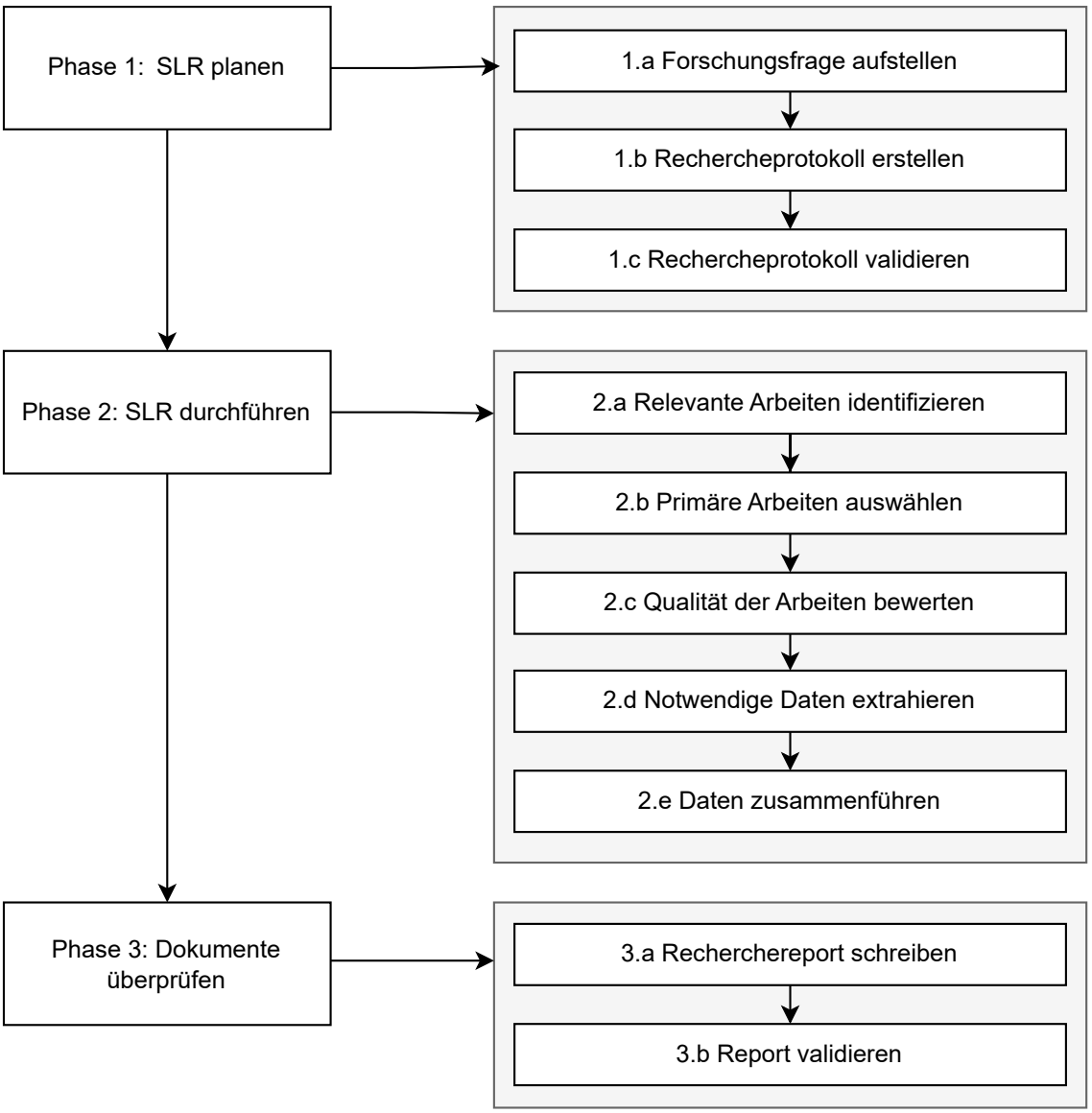

Abbildung 3.1: Ablauf der systematischen Literaturrecherche nach Brereton et al. [BKB+07]

Forschungsfrage: Wie können bestehende Low-Code-Entwicklungsansätze, insbesondere im Kontext von Model-Driven Engineering (MDE) und Open-Source-Prinzipien, für die Anforderungen der Quantencomputing-Anwendungsentwicklung genutzt werden? Falls keine spezifischen Low-Code-Werkzeuge für Quantencomputing existieren, welche klassischen Low-Code-Werkzeuge bieten Potenzial für eine solche Integration?

Für die Literaturrecherche wurden die wissenschaftlichen Datenbanken *Google Scholar*, *IEEE Xplore* und *ACM Digital Library* genutzt. Die Suchstrategie wurde gezielt entwickelt, um eine umfassende Abdeckung der relevanten Themenbereiche zu gewährleisten.

Bei der Auswahl der Suchbegriffe wurde darauf geachtet, die Breite der Themenfelder abzudecken. Die Begriffe wurden basierend auf ihrer Relevanz für die Forschungsfrage ausgewählt und decken die zentralen Aspekte der Arbeit ab: *Low-Code Development*, *Quantum Computing*, *Model-Driven Engineering* und *Open Source*.

Im Rahmen der explorativen Suchen wurden diese Suchbegriffe in verschiedenen Kombinationen verwendet. Dabei kamen die Boolesche Operatoren *AND* und *OR* zum Einsatz, um die Suche zu präzisieren und relevante Literatur gezielt zu identifizieren.

Um eine hohe Aktualität und Relevanz der gefundenen Studien sicherzustellen, wurden nur Veröffentlichungen berücksichtigt, die ab dem Jahr 2014 erschienen sind. Dieses Kriterium hilft sicherzustellen, dass die untersuchten Tools und Methoden noch aktiv entwickelt und gewartet werden, was insbesondere im schnelllebigen Bereich des Quantencomputing von Bedeutung ist.

Zusätzlich wurden folgende Einschlusskriterien festgelegt: Es werden ausschließlich Studien berücksichtigt, die in englischer oder deutscher Sprache verfasst sind und ein Peer-Review-Verfahren durchlaufen haben. Diese methodische Herangehensweise gewährleistet eine solide und wissenschaftlich fundierte Grundlage für die Analyse und Entwicklung von Low-Code-Ansätzen für Quantencomputing-Anwendungen.

Aufgrund des zeitlich begrenzten Umfangs dieser Arbeit konnte nur eine eingeschränkte Anzahl an Publikationen gesichtet werden. Es wurde jedoch darauf geachtet, dass die Einschränkungen so gesetzt sind, dass möglichst wenige relevante Publikationen ausgeschlossen werden. Die Auswahlkriterien wurden sorgfältig gewählt, um sicherzustellen, dass die Ergebnisse trotz der Einschränkungen eine hohe Relevanz und Aussagekraft für die Forschungsfragen dieser Arbeit behalten.

Die Relevanz der Publikationen wurde anhand eines gestuften Auswahlverfahrens bestimmt: Zunächst wurden die Titel gesichtet, um eine erste Einschätzung hinsichtlich der thematischen Passgenauigkeit zu gewinnen. Anschließend wurden die Abstracts der verbleibenden Publikationen detailliert analysiert, um deren inhaltliche Relevanz und Beitrag zu den Forschungsthemen der Arbeit zu bewerten. Nur Publikationen, die sowohl im Titel als auch im Abstract eine hohe Übereinstimmung mit den definierten Kriterien aufwiesen, wurden in die engere Auswahl aufgenommen.

ResearchRabbit

In dieser Arbeit wird ResearchRabbit als ergänzendes Tool zur systematischen Literaturrecherche eingesetzt. ResearchRabbit ist ein Softwaretool, das entwickelt wurde, um Arbeiten aus Forschungsgebieten, basierend auf einer Suchanfrage des Nutzers, als Graph darzustellen und relevante Verbindungen zwischen wissenschaftlichen Publikationen auf Basis von Zitierungen und thematischen Ähnlichkeiten zu visualisieren [CB23]. Der erstellte Graph bietet eine visuelle Darstellung, die es ermöglicht, zentrale Arbeiten und bisher weniger beachtete Verbindungen schnell zu identifizieren. Durch diese Eigenschaften eignet sich ResearchRabbit als Unterstützung für den Schritt des Snowballings in der systematischen Literaturrecherche.

Die Anwendung von ResearchRabbit in der systematischen Literaturrecherche bietet erhebliche Vorteile. Erstens steigert das Tool die Effizienz des Rechercheprozesses, indem es durch die Visualisierung von Forschungsnetzwerken relevante Studien schneller erkennbar macht, was den Zeitaufwand für das erste Screening reduziert. Zweitens erweitert ResearchRabbit die Möglichkeit, wichtige Arbeiten zu entdecken, die in traditionellen Datenbankensuchen möglicherweise übersehen werden würden. Dies geschieht insbesondere durch die Aufdeckung von Querverbindungen zwischen verschiedenen Forschungsgebieten. Darüber hinaus ermöglicht die Analyse von Netzwerkbeziehungen tiefere Einblicke in die Einflussnahme und den thematischen Kontext von Schlüsselpublikationen, was zu einem besseren Verständnis des Forschungsstandes beiträgt. Schließlich erlaubt das Tool, aktuelle Forschungsentwicklungen zu verfolgen und die Literaturrecherche regelmäßig mit neuen relevanten Publikationen zu aktualisieren.

Die Entscheidung, ResearchRabbit einzusetzen, basiert auf der Notwendigkeit, in einer umfangreichen und dynamischen Forschungslandschaft effizient zu navigieren. In sich schnell weiterentwickelnden und interdisziplinären Feldern wie Low-Code Entwicklung und Quantencomputing bietet ResearchRabbit einen signifikanten Mehrwert, indem es komplexe Informationsstrukturen zugänglich und handhabbar macht. Durch die Integration dieses Tools in die systematische Literaturrecherche wird nicht nur die Effizienz des Prozesses verbessert, sondern auch die Qualität und Tiefe der Forschungsergebnisse erhöht.

3.2 Durchführung des SLR

Basierend auf den im vorherigen Abschnitt festgelegten Kriterien wird das SLR durchgeführt. Im Folgenden werden die einzelnen Schritte der Recherche detailliert dokumentiert und die Ergebnisse visualisiert.

Eine initiale explorative Suche in Google Scholar mit der Suchanfrage `intitle: ‘‘low-code’’ tools platforms comparison overview` lieferte 489 Ergebnisse. Diese werden explorativ gesichtet und auf Relevanz geprüft. Insgesamt wurden initial vier Publikationen als relevant identifiziert. Zwei davon sind insbesondere relevant, da sie einen Vergleich verschiedener Low-Code-Plattformen und deren Charakteristika ziehen.

In einer ersten unstrukturierten Suche wurden zudem zwei Publikationen zu MDE und Low-Code Entwicklung aus dem Quantencomputing und aus dem Open Source Bereich identifiziert und hinzugenommen. Diese sind zum einen Publikation über *Classiq* [Min22], zum anderen eine Publikation über *BESSER* [ACS+24]. Classiq ist ein Framework zur Entwicklung von Quantencomputeralgorithmen, das darauf abzielt, die Komplexität der Programmierung auf Gatterniveau zu reduzieren und die Zugänglichkeit für Entwickler zu verbessern. Es ermöglicht die Erstellung optimierter und an die Hardware angepasster Quantenschaltkreise aus abstrakten funktionalen Modellen. Dieses Framework automatisiert den Prozess der Quantenalgorithmuserstellung, indem es Nutzern erlaubt, sich auf die funktionalen Anforderungen zu konzentrieren, während das Framework die Details der Schaltkreisimplementierung übernimmt. Ein Hauptvorteil von Classiq ist die Möglichkeit, Quantenalgorithmen visuell zu entwerfen und automatisch die besten Implementierungsoptionen zu ermitteln, die den Systemanforderungen und Hardwarebeschränkungen entsprechen. Dies soll den Aufwand für handgeschriebenen Code und Optimierung reduzieren [Min22].

Das BESSER-Framework ist eine Open-Source Low-Code-Plattform. Ein Merkmal von BESSER ist seine Erweiterbarkeit und Anpassungsfähigkeit. Die Plattform ist so konzipiert, dass sowohl die Spezifikationsmethoden als auch die Code-Generatoren von der Community erweitert und angepasst werden können. Dies fördert nicht nur die Innovation innerhalb der Entwicklergemeinschaft, sondern ermöglicht es auch, spezifische Anforderungen und Präferenzen der Nutzer zu berücksichtigen. Da BESSER eine Open-Source-Plattform ist, können Nutzer die Plattform frei modifizieren und erweitern, was die Gefahr eines Vendor-Lock-ins minimiert und die Flexibilität erhöht. Darüber hinaus adressiert BESSER die wachsende Komplexität moderner Softwaresysteme, indem es Low-Code-Entwicklungsansätze mit fortschrittlichen Modellierungs- und Generierungstechniken kombiniert. Dies macht es zu einem geeigneten Werkzeug für die Entwicklung komplexer, intelligenter Softwarelösungen, die auf die Bedürfnisse einer breiten Nutzerbasis zugeschnitten sind, von technischen Experten bis hin zu Geschäftsanwendern [ACS+24].

Protokollbasierte Suche

Im Schritt der protokollbasierten Suche werden die vorgestellten Suchbegriffe und Kriterien für die systematische Literaturrecherche angewandt. In der ersten Iteration der protokollbasierten Suche wird die folgende Suchanfrage in den Datenbanken verwendet:

Suchanfrage 1 (Durchgeführt am: 12.07.2024):

```
‘‘Low-Code Development’’ OR ‘‘Quantum Computing’’ OR ‘‘Model-Driven
Engineering’’ OR ‘‘Open Source’’
```

Tabelle 3.1: Anzahl der Suchergebnisse für Suchanfrage 1

Datenbank	**Anzahl der Ergebnisse**
Google Scholar	17.800
IEEE Xplore	62.421
ACM Digital Library	79.035

Diese Suchanfrage lieferte die in Tabelle 3.1 dargestellten Ergebnisse. Die große Anzahl an Ergebnissen zeigt die große Bandbreite und Relevanz der Themenfelder. Um zusätzlich einen Überblick zu gewinnen, wie die Verteilung der Publikationen zu den jeweiligen Konzepten ausfällt, wird eine Analyse der Anzahl der Publikationen pro Jahr und Konzept, also MDE oder Low-Code vorgenommen. Die Ergebnisse dieser Analyse sind in den Grafiken 3.2 und 3.3 dargestellt. In der Grafik 3.3 ist die x-Achse für das Erscheinungsjahr bewusst auch außerhalb des vorher gesteckten Zeitfilters angelegt, um zu verdeutlichen, dass LCD im Vergleich zu MDE ein deutlich jüngeres Forschungsthema ist.

Hier fällt auf, dass die Anzahl der Publikationen zu Low-Code-Entwicklung erstens insgesamt niedriger ausfällt als die Anzahl der Publikationen zu Model-Driven Engineering. Da die Anzahl der Ergebnisse insgesamt zu groß ist, um effektiv bearbeitet zu werden, wurde die Suche angepasst. Außerdem fällt bei einer ersten Durchsicht auf, dass viele Ergebnisse nicht die gewünschte Schnittmenge aus Low-Code-Entwicklung und Quantencomputing abdecken, sondern nur eines der beiden Themenfelder behandeln. Um also die Relevanz der Ergebnisse zu erhöhen, werden spezifischere Suchanfragen formuliert.

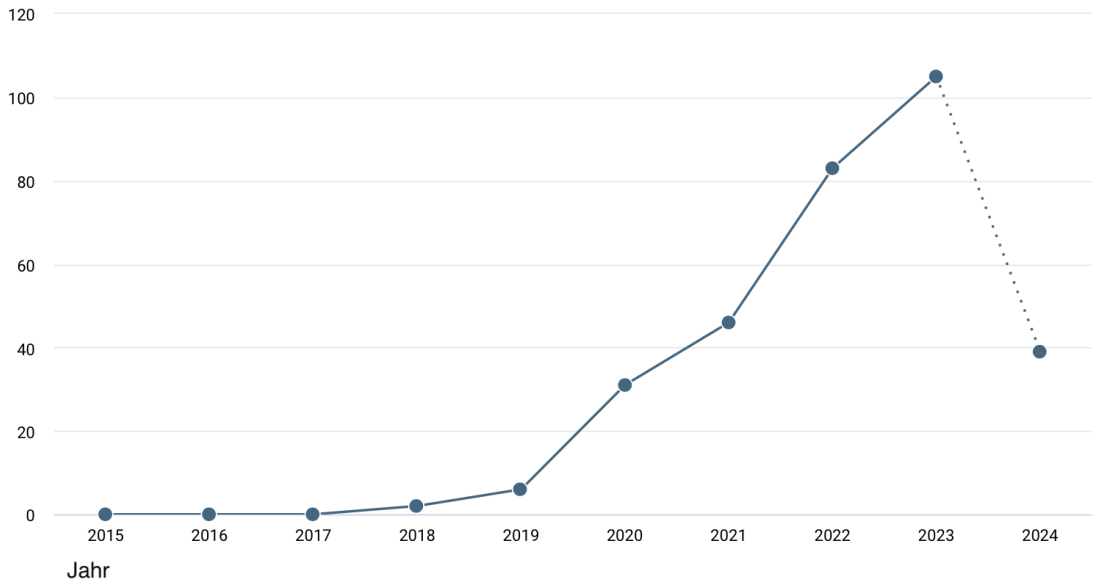

Abbildung 3.2: Anzahl der Publikationen LCD pro Jahr nach Suchanfrage 1

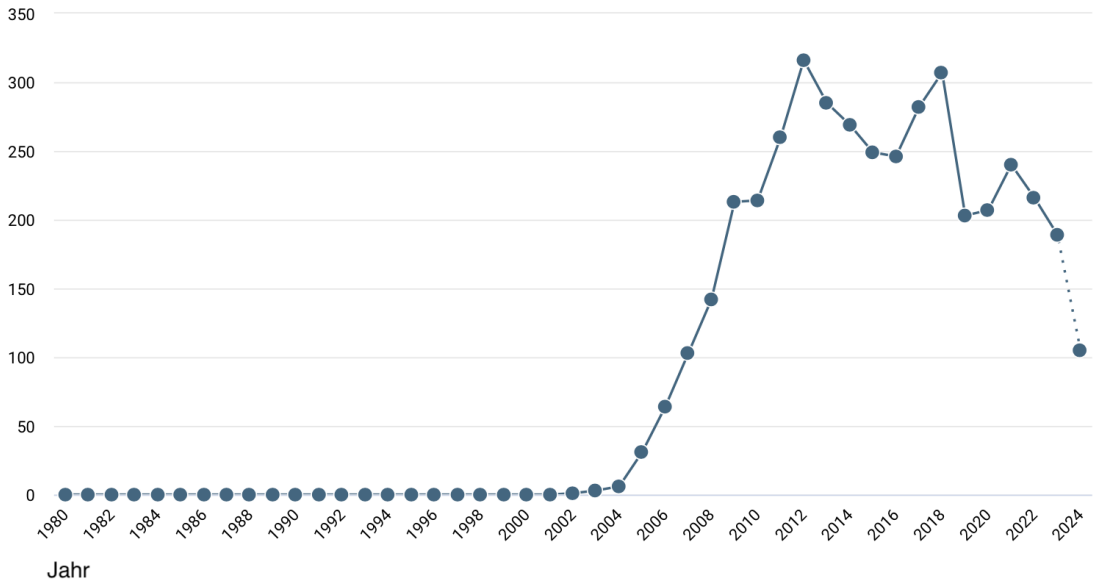

Abbildung 3.3: Anzahl der Publikationen zu MDE pro Jahr nach Suchanfrage 1

In der zweiten Iteration der protokollbasierten Suche wurde die folgende Suchanfrage verwendet:

Suchanfrage 2 (Durchgeführt am: 14.07.2024):

```
(‘‘Low Code Platform’’ OR ‘‘Low Code Development’’ OR ‘‘Low-Code Development’’
OR ‘‘Model-Driven Engineering’’) AND (‘‘Quantum Computing’’)
```

Tabelle 3.2: Anzahl der Suchergebnisse für Suchanfrage 2

Datenbank	Anzahl der Ergebnisse
Google Scholar	176
IEEE Xplore	3
ACM Digital Library	30

Die Ergebnisse der zweiten Suchanfrage sind in Tabelle 3.2 dargestellt. Im Hinblick auf die Erscheinungsjahre fällt auf, dass die drei Publikationen in IEEE Xplore zwischen 2021 und 2023 erschienen. In der ACM Digital Library sind die Publikationen zwischen 2021 und 2024 erschienen. Lediglich die Google Scholar Suche ergab Publikationen, die bis ins Jahr 2014 als Untergrenze der gewünschten Filterung zurückreichen.

Da die Anzahl der Ergebnisse nun sinnvoller zu bearbeiten war, wurde mit der Durchsicht der Titel und Abstracts fortgefahren. Die Durchsicht der Abstracts ergab, dass die Publikationen aus IEEE Xplore alle drei relevant sind, da sie sich inhaltlich in der Schnittmenge aus MDE und Quantencomputing befinden. Als Nächstes wurden beim Schritt der Auswahl notwendigerweise die Ergebnisse der ACM Digital Library gezielt gefiltert, um die Relevanz und Qualität der eingeschlossenen Studien sicherzustellen. Von den initialen 30 Ergebnissen aus der ACM Digital Library erfüllten lediglich vier die Auswahlkriterien. Diese strenge Selektion ist entscheidend, um sicherzustellen, dass die verbleibenden Studien tatsächlich die Forschungsfrage 3.1 adressieren und qualitative Erkenntnisse liefern können. Aus den 176 Ergebnisse aus Google Scholar erfüllten 20 Publikationen die Kriterien für die engere Auswahl. Im Schritt der Deduplizierung wurden die Ergebnisse aus den drei Datenbanken zusammengeführt. So ergaben sich ohne Duplikate insgesamt 25 Publikationen, die für die weitere Analyse in Betracht gezogen werden.

Bevor durch Snowballing weitere möglicherweise relevante Publikationen gesucht wurden und die Volltexte der Publikationen analysiert werden, wurde eine dritte Suchanfrage formuliert. Diese zielte darauf ab, durch eine angepasste Formulierung der Anfrage weitere bisher nicht gefundene Ergebnisse zu identifizieren. Es zeigte sich, dass die Formulierung `"low-code"` mit Bindestrich ausreichend ist, da Publikationen, die `low code` ohne Bindestrich enthalten, auch in den Ergebnissen enthalten sind und die Schreibweise mit Bindestrich gängiger ist.

Suchanfrage 3 (Ausgeführt am: 16.07.2024):

```
(intitle:‘‘low-code’’ quantum computing)
```

Hierfür ergibt sich die in Tabelle 3.3 dargestellte Anzahl an Ergebnissen.

Tabelle 3.3: Anzahl der Suchergebnisse für Suchanfrage 3

Datenbank	Anzahl der Ergebnisse
Google Scholar	26
IEEE Xplore	1
ACM Digital Library	8

Durchsicht und Deduplikation der neuen Ergebnisse ergab, dass zwar aus Google Scholar keine weiteren Publikationen für die engere Auswahl übernommen werden konnten, jedoch aus der ACM Digital Library sieben sowie aus IEEE Xplore eine Publikation. Insgesamt ergibt sich somit eine Anzahl von 33 unterschiedlichen Publikationen, die für die weitere Analyse in Betracht gezogen werden.

Mit den nun identifizierten Publikationen wird der Schritt des Snowballings durchgeführt. Hierbei wird ResearchRabbit eingesetzt, um weitere Publikationen zu identifizieren, die inhaltlich mit den bereits identifizierten Publikationen in Verbindung stehen und somit potenziell relevant sind. Das Tool zeigt dabei die Verbindungen zwischen den Publikationen an und ermöglicht es, die Netzwerke zu älteren und neueren Publikationen zu visualisieren. Es ergeben sich 21 ältere Publikationen, die mit den bereits identifizierten 33 Publikationen in Verbindung stehen. Dazu kommen sechs neuere die auf die bereits identifizierten Publikationen verweisen.

Für diese wurde ebenfalls der Titel und Abstract durchgesehen, um zu entscheiden, ob sie in die engere Auswahl übernommen werden. Insgesamt ergaben sich nach erneuter Durchsicht so insgesamt 41 Publikationen aus explorativer und protokollbasierter Suche. Diese sind in der Tabelle A1 dargestellt.

Die Metadaten der resultierenden Menge von Publikationen wurde mithilfe von Excel untersucht. Dabei wurde für tiefergehende Einblicke in die thematischen Schwerpunkte der Publikationen das Abstract der Arbeit nach Schlüsselworten untersucht, um sie jeweils den entsprechenden Kategorien zuzuordnen. Hierbei kam die Formel 3.1 zum Einsatz.

```
=CONCAT(
    IF(OR(ISNUMBER(SEARCH("Low"; D12)); ISNUMBER(SEARCH("Low"; E12))); "Low-Code, "; "");
    IF(OR(ISNUMBER(SEARCH("Quantum"; D12)); ISNUMBER(SEARCH("Quantum"; E12))); "Quantum
      Computing, "; "");
    IF(OR(ISNUMBER(SEARCH("model"; D12)); ISNUMBER(SEARCH("model"; E12))); "Model-Driven
      Engineering, "; "");
    IF(OR(ISNUMBER(SEARCH("Open Source"; D12)); ISNUMBER(SEARCH("Open Source"; E12)));
      "Open Source"; "")
    )
```

Listing 3.1: Excel Formel zur Suche von Worten im Abstract

Die Formel generiert je nach Schlüsselwort im Abstract einen String, der Aufschluss über die thematischen Schwerpunkte der Publikation gibt. Da nur das Abstract und nicht die jeweilige gesamte Ausarbeitung für die Suche nach den Schlüsselworten genutzt wurde, ist die Präzision der Zuordnung nicht so hoch wie unter Berücksichtigung des gesamten Textes der Publikation. Dieser Kompromiss aus Schnelligkeit und Genauigkeit wurde bewusst in Kauf genommen, um die Effizienz des Prozesses zu erhöhen und dabei dennoch eine Übersicht über die grobe Verteilung der Themenbereiche zu erhalten. Die Tabelle A1 aller Publikationen inklusive der jeweiligen thematischen Einordnung ist im Anhang der Ausarbeitung. Die zeitliche Verteilung der Publikationen zeigt sich über die Jahre hinweg unterschiedlich stark. Abbildung 3.4 veranschaulicht, wie sich die Anzahl der relevanten Publikationen im Verlauf der Jahre entwickelt hat. Die Kategorisierung der Publikationen nach ihren thematischen Schwerpunkten, dargestellt in Abbildung 3.5, visualisiert die Häufigkeit der verschiedenen Themenbereiche und ermöglicht eine Einsicht in die Verteilung der Forschungsschwerpunkte innerhalb des Low-Code- und Quantencomputing-Feldes.

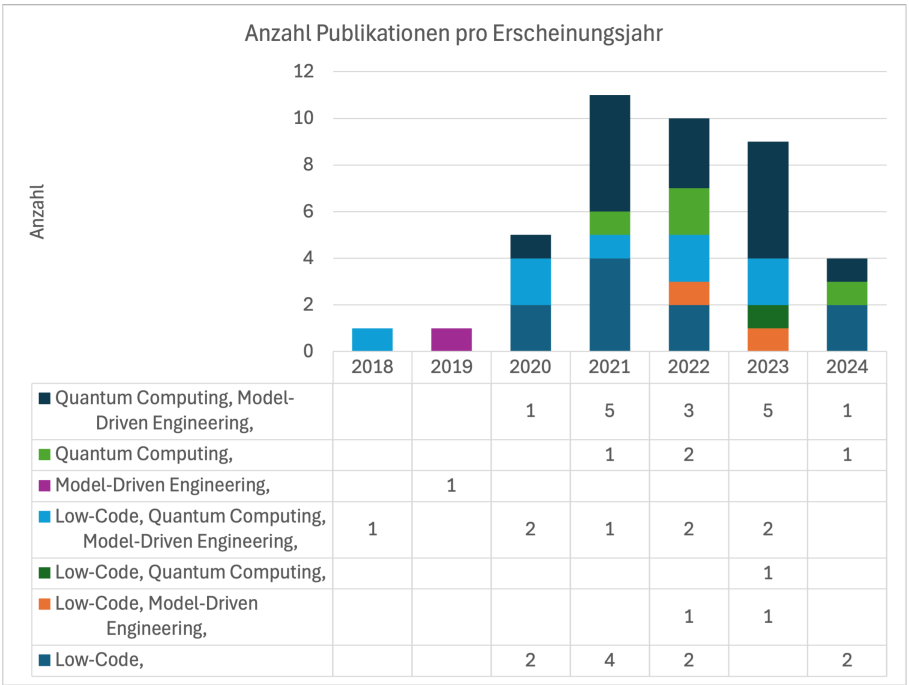

Abbildung 3.4: Anzahl der Publikationen pro Jahr

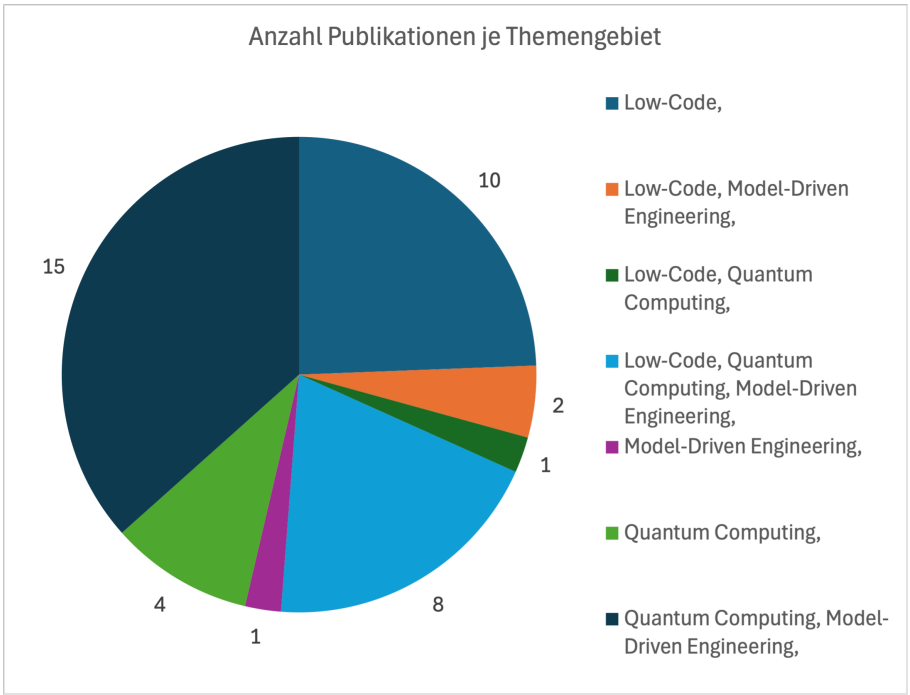

Abbildung 3.5: Anzahl der Publikationen pro Themenbereich

3.3 Synthese und Bewertung der Ergebnisse

Die Synthese und Bewertung der Ergebnisse erfolgt in mehreren Schritten, um die identifizierten Studien zu analysieren, zu interpretieren und zu bewerten. Dabei werden die Hauptbefunde zusammengefasst, die Erkenntnisse integriert und Forschungslücken identifiziert.

Vor dem Hintergrund der gestellten Forschungsfrage 3.1 werden Kriterien aufgestellt, anhand derer die Publikationen bewertet werden. In Tabelle 3.4 sind die Kriterien dargestellt, die für die Forschungsfrage herangezogen werden.

Tabelle 3.4: Kriterien zur Bewertung der Publikationen

Kriterium	**Beschreibung**
Low-Code Tools	Welche Low-Code Tools, Plattformen oder Konzepte stehen im Zentrum der Publikation?
Quantencomputing	Werden Anpassbarkeit der Plattformen (auf Quantencomputing) erwähnt, und falls ja, wie?
Aktualität	Wie aktuell und zukunftssicher sind die in der Publikation behandelten Plattformen und wie relevant sind sie in der Praxis?
Open-Source	Sind die erwähnten Ansätze frei zugänglich oder kommerziell?

Low-Code Tools

Die Untersuchung der in den Publikationen beschriebenen Low-Code-Werkzeuge zeigt eine klare Ausrichtung auf die Verbesserung der Effizienz und Zugänglichkeit in der Softwareentwicklung, jedoch mit unterschiedlichen Schwerpunkten und Anwendungsbereichen. Die Arbeiten von Sahay et al. [SSI+20] und Bock [BBFF21a] analysieren umfassend kommerzielle Low-Code-Plattformen wie OutSystems, Mendix, Appian, PowerApps und Salesforce. Diese Plattformen wurden primär für klassische Softwareentwicklungsprozesse entwickelt und zeichnen sich durch Benutzerfreundlichkeit sowie die Fähigkeit zur Integration in bestehende IT-Systeme aus. Die systematische Literaturübersicht von Pinho und Amaral [PPA+22] ergänzt diese Analyse, indem sie sich auf die Benutzerfreundlichkeit von Low-Code-Plattformen fokussiert, allerdings ohne spezifische Tools zu nennen.

Ein differenzierteres Bild bietet die Arbeit von Rao et al. [RTK+24], die die Integration von Künstlicher Intelligenz (KI) in Low-Code-Plattformen untersucht. Hier wird gezeigt, wie Low-Code-Ansätze die Entwicklung von KI-Modellen vereinfachen und für eine breitere Benutzergruppe zugänglich machen können. Diese Perspektive erweitert den Anwendungsbereich von Low-Code-Plattformen über traditionelle Softwareentwicklungsprozesse hinaus.

Ein besonders relevantes Beispiel im Kontext von Quantencomputing ist die Plattform Quantumoonlight, beschrieben von Amato und Cicalese [ACC+23a]. Quantumoonlight ist speziell für Quantum Machine Learning (QML) entwickelt worden und stellt eine benutzerfreundliche Umgebung bereit,

in der Quantenalgorithmen modelliert und getestet werden können. Diese Plattform, die sowohl als kommerzielle als auch als Open-Source-Lösung verfügbar ist, zeigt das Potenzial spezialisierter Low-Code-Werkzeuge für das Quantencomputing.

Darüber hinaus untersuchen Moin et al. [MBC23] die Anwendung von Model-Driven Engineering (MDE) im Rahmen der Low-Code-Entwicklung, insbesondere für Quantum Federated Learning (QFL). Diese Arbeit verdeutlicht, wie MDE-Tools, die ursprünglich für maschinelles Lernen entwickelt wurden, modifiziert werden können, um Quantenanwendungen zu unterstützen, und unterstreicht das Potenzial von MDE als integralen Bestandteil der Low-Code-Entwicklung im Quantencomputing-Bereich.

Insgesamt zeigt die Analyse, dass die meisten untersuchten Low-Code-Plattformen für klassische Softwareentwicklungsprozesse entwickelt wurden, während Anwendungen im Bereich des Quantencomputings bisher eine untergeordnete Rolle spielen. Quantumoonlight stellt eine bemerkenswerte Ausnahme dar und verdeutlicht, dass spezialisierte Low-Code-Werkzeuge für das Quantencomputing möglich und vielversprechend sind. Diese Forschungslücke weist darauf hin, dass es notwendig ist, Low-Code-Ansätze weiterzuentwickeln, um sie für die Anforderungen des Quantencomputings zu adaptieren und damit eine breitere Anwendung dieser Technologie zu ermöglichen.

Zusammengefasst sind die Plattformen in der Tabelle 3.5 aufgeführt.

Plattform/Tool	**Erwähnte Publikationen**
OutSystems	Sahay et al. [SSI+20], Bock [BBFF21a], Prinz und Huber [PRH21], Khorram et al. [KKM+20]
Mendix	Sahay et al. [SSI+20], Bock [BBFF21a], Prinz und Huber [PRH21], Khorram et al. [KKM+20]
Appian	Sahay et al. [SSI+20], Bock [BBFF21a], Khorram et al. [KKM+20]
PowerApps	Sahay et al. [SSI+20], Bock [BBFF21a]
Salesforce	Sahay et al. [SSI+20], Bock [BBFF21a], Prinz und Huber [PRH21]
Betty Blocks	Bock [BBFF21a]
GeneXus	Bock [BBFF21a]
Quick Base	Bock [BBFF21a]
MontiAnna und ML-Quadrat	Moin et al. [MBC23]
Quantumoonlight	Amato und Cicalese [ACC+23a]

Tabelle 3.5: Behandelte Low-Code-Plattformen und Tools in den untersuchten Publikationen

Quantencomputing

Zunächst zeigt die Arbeit von Sahay et al. [SSI+20] eine gründliche Untersuchung traditioneller Low-Code-Plattformen wie OutSystems und Mendix, die zwar primär für klassische Softwareentwicklungsprozesse entwickelt wurden, jedoch aufgrund ihrer architektonischen Modularität gewisse Möglichkeiten zur Erweiterung auf neue Technologien bieten könnten. Die Plattformen zeichnen sich durch eine hohe Anpassungsfähigkeit und Integration in bestehende Systeme aus,

was theoretisch die Grundlage für eine Erweiterung um Quantencomputing-Funktionen bilden könnte. Ein Hindernis bleibt jedoch die Tatsache, dass diese Plattformen stark auf proprietäre Lösungen setzen, was die Implementierung von Open-Source-Quantum-APIs oder SDKs erschweren könnte.

Die Analyse von Bock [BBFF21a] ergänzt dies, indem sie auf die Kernmerkmale und Einschränkungen weiterer Plattformen eingeht, darunter auch Plattformen wie Betty Blocks und GeneXus, die aufgrund ihrer Flexibilität und spezifischen Anwendungsfälle theoretisch für Erweiterungen in Richtung Quantencomputing geeignet erscheinen. Besonders die Möglichkeit zur Nutzung von APIs und der bereits existierende Support für verschiedenste Technologien könnten hier von Vorteil sein. Allerdings bleibt auch in dieser Arbeit der Fokus auf traditionellen Anwendungsfällen, ohne konkrete Beispiele für eine Erweiterung auf Quantencomputing.

Pinho und Amaral [PPA+22] bieten eine Perspektive, die sich ausschließlich auf die Benutzerfreundlichkeit konzentriert, was für eine Erweiterung auf komplexere Anwendungsbereiche wie Quantencomputing sowohl ein Vorteil als auch eine Herausforderung sein kann. Plattformen, die für hohe Benutzerfreundlichkeit optimiert sind, könnten aufgrund ihrer intuitiven Bedienung theoretisch leichter an neue, anspruchsvollere Technologien angepasst werden. Andererseits könnte die Vereinfachung, die oft mit Benutzerfreundlichkeit einhergeht, die Integration sehr spezifischer und komplexer Quantencomputing-Komponenten erschweren.

Rao et al. [RTK+24] untersuchen die Integration von KI in Low-Code-Plattformen, was interessante Parallelen zu möglichen Erweiterungen für Quantencomputing bietet. Die Fähigkeit dieser Plattformen, KI-Modelle zu integrieren und komplexe Prozesse zu vereinfachen, deutet darauf hin, dass ähnliche Anpassungen für Quantencomputing denkbar wären. Dies setzt allerdings voraus, dass die Plattformen ausreichend flexibel sind, um nicht nur klassische Technologien, sondern auch die spezifischen Anforderungen des Quantencomputings zu berücksichtigen, etwa durch die Implementierung spezieller Gate-Modelle oder Quanten-Simulatoren.

Die Arbeit von Amato und Cicalese [ACC+23a] hebt sich dadurch ab, dass sie explizit eine Low-Code-Plattform für Quantencomputing, Quantumoonlight, vorstellt. Diese Plattform zeigt, dass es durchaus möglich ist, Low-Code-Ansätze auf Quantencomputing-Anwendungen zu erweitern. Quantumoonlight bietet Funktionen zur Modellierung und Implementierung von Quantum Machine Learning (QML) und demonstriert, wie eine Plattform durch geeignete Erweiterungen speziell auf die Anforderungen des Quantencomputings angepasst werden kann. Diese spezifische Anpassung ist beispielhaft für die Art von Erweiterbarkeit, die auch bei anderen Low-Code-Plattformen theoretisch realisiert werden könnte, wenn sie mit ähnlichen Tools und APIs ausgestattet werden.

Moin et al. [MBC23] zeigen mit ihrer Untersuchung zur Erweiterung von Model-Driven Engineering (MDE) für Quantum Federated Learning (QFL), dass bestehende MDE-Tools durch gezielte Anpassungen für Quantenanwendungen nutzbar gemacht werden können. Diese Arbeit unterstreicht die Möglichkeit, bestehende technologische Frameworks zu erweitern, um sie für neue, komplexere Anwendungsbereiche nutzbar zu machen. Das Beispiel des QFL verdeutlicht, wie solche Erweiterungen spezifische Anforderungen des Quantencomputings berücksichtigen können, indem sie etwa die Komplexität durch Abstraktion und Automatisierung reduzieren.

In der Arbeit von Pérez-Delgado und Pérez-González [PPPP20] sowie in den Untersuchungen von Gemeinhardt et al. [GGW21b] [GGWW18] wird das Potenzial zur Erweiterung bestehender Modellierungsansätze durch die Einführung quantenspezifischer Modelle und Frameworks beleuchtet. Die vorgeschlagenen Erweiterungen bestehender Modellierungssprachen wie UML zeigen auf, dass

es möglich ist, bekannte und weit verbreitete Softwareentwicklungspraktiken auf den Bereich des Quantencomputings zu übertragen. Solche Erweiterungen könnten auch für Low-Code-Plattformen relevant werden, die durch die Integration von Modellierungstools, die Quantenlogik unterstützen, auf neue Anwendungsbereiche ausgeweitet werden könnten.

Insgesamt lässt sich feststellen, dass viele der untersuchten Arbeiten implizit oder explizit Hinweise darauf geben, wie bestehende Low-Code-Plattformen und Modellierungsansätze an die Anforderungen des Quantencomputings angepasst werden könnten. Diese Anpassungen erfordern jedoch eine bewusste Erweiterung der Plattformen um spezifische Quantenkomponenten und -funktionen, was nicht trivial ist und sorgfältige Planung und Umsetzung voraussetzt. Dies stellt nicht nur technische, sondern auch organisatorische Herausforderungen dar, insbesondere in Bezug auf die Integration von Quanten-APIs, die Schulung der Nutzer und die Anpassung der Plattformen an die komplexe Natur des Quantencomputings.

Aktualität

Im Rahmen der Bewertung der Aktualität und Nachhaltigkeit der in den Publikationen erwähnten Low-Code-Plattformen zeigt sich eine differenzierte Landschaft. OutSystems, Mendix, Appian, PowerApps und Salesforce zählen zu den führenden kommerziellen Plattformen, die sich seit mehreren Jahren auf dem Markt etabliert haben. Diese Plattformen werden kontinuierlich weiterentwickelt und zeichnen sich durch eine aktive Community sowie regelmäßige Updates aus, was ihre langfristige Relevanz und Anpassungsfähigkeit an neue technologische Entwicklungen unterstreicht. Dies macht sie zu zuverlässigen Optionen für Unternehmen, die auf bewährte und gepflegte Lösungen setzen möchten.

Im Gegensatz dazu weist Quantumoonlight, obwohl es speziell für Quantum Machine Learning entwickelt wurde und eine vielversprechende Nische adressiert, deutliche Anzeichen einer eingeschränkten Weiterentwicklung auf. Mit dem letzten Update vor einem Jahr und nur wenigen GitHub-Sternen deutet dies auf eine geringere Community-Unterstützung und möglicherweise eingeschränkte Zukunftsfähigkeit hin. Diese Faktoren werfen Fragen hinsichtlich der langfristigen Relevanz und der Fähigkeit der Plattform auf, sich an zukünftige Entwicklungen im Quantencomputing anzupassen.

Plattformen wie Betty Blocks, GeneXus und Quick Base zeigen ähnliche Charakteristika wie die führenden kommerziellen Lösungen, obwohl sie nicht die gleiche Marktdurchdringung oder Community-Größe wie OutSystems oder Mendix besitzen. Diese Tools profitieren jedoch ebenfalls von regelmäßigen Updates und einem soliden Ruf in spezifischen Anwendungsbereichen, was ihre Relevanz und Nachhaltigkeit stützt.

Die Plattformen, die in spezifischeren Kontexten wie der KI-Entwicklung oder der Quantensoftware-Entwicklung genannt werden, zum Beispiel in Arbeiten wie Quantumoonlight, spiegeln häufig innovative Ansätze wider, sind jedoch in ihrer Marktpräsenz und der Breite der Community-Akzeptanz variabel. Die Nachhaltigkeit dieser Plattformen hängt stark von der fortlaufenden Entwicklung und der Akzeptanz durch die Nutzerbasis ab, was bei einigen dieser Plattformen derzeit fraglich erscheint.

Diese Analyse zeigt, dass während einige Low-Code-Plattformen durch fortlaufende Weiterentwicklung und starke Marktpräsenz glänzen, andere, besonders in Nischenbereichen wie Quantencomputing, ihre langfristige Relevanz noch unter Beweis stellen müssen.

Open-Source

Die Plattformen OutSystems, Mendix, Appian, PowerApps und Salesforce sind alle kommerzielle Produkte, die eine kostenpflichtige Lizenz erfordern, um ihre volle Funktionalität zu nutzen. Diese Plattformen sind auf den kommerziellen Markt ausgerichtet und bieten umfassende, aber proprietäre Entwicklungsumgebungen an [SSI+20]. Ebenso sind Betty Blocks, GeneXus und Quick Base kommerzielle Plattformen, die primär für Unternehmenskunden entwickelt wurden und ähnliche Lizenzanforderungen haben [BBFF21a].

Im Gegensatz dazu sind Quantumoonlight und einige der in der Forschung zu Model-Driven Engineering (MDE) für Quantum Computing diskutierten Ansätze, wie die Erweiterungen für UML-Modelle, auch als Open-Source-Lösungen verfügbar. Quantumoonlight, das speziell für Quantum Machine Learning entwickelt wurde, ermöglicht es Entwicklern, die Plattform ohne kommerzielle Lizenzkosten zu nutzen, was ihre Zugänglichkeit und Anpassungsfähigkeit erhöht [ACC+23a]. Ähnlich bieten die im Zusammenhang mit MDE entwickelten Open-Source-Erweiterungen für UML eine flexible und erweiterbare Grundlage für die Modellierung von Quantenanwendungen [GGW21b]. Diese Open-Source-Lösungen sind besonders wertvoll für Forscher und Entwickler, da sie die Weiterentwicklung und Anpassung der Technologien erleichtern und die Barrieren für den Zugang zu diesen innovativen Ansätzen senken.

Gemeinsame Trends und Muster in den identifizierten Studien

Vor dem Hintergrund der Forschungsfrage 3.1, wie Low-Code-Entwicklungsansätze effektiv in die Konzeption und Entwicklung von Quantencomputing-Anwendungen integriert werden können, unter besonderer Berücksichtigung von Model-Driven Engineering (MDE) und Open-Source-Prinzipien, lassen sich mehrere gemeinsame Trends und Muster in den identifizierten Studien erkennen.

Ein wiederkehrendes Thema ist die wachsende Bedeutung von Low-Code-Plattformen zur Vereinfachung und Beschleunigung der Entwicklungsprozesse, die insbesondere durch visuelle Entwicklungsumgebungen und die Minimierung manuellen Codierens erreicht wird [BBFF21b; KKM+20; SSI+20]. Diese Eigenschaften sind besonders relevant, um die komplexen Konzepte des Quantencomputings einem breiteren Publikum zugänglich zu machen.

Ein weiterer Trend ist die Integration von Model-Driven Engineering (MDE) in Low-Code-Plattformen, um die Effizienz und Produktivität zu steigern [GGG+21; MMC+21; PPP+22]. Dies könnte die Entwicklung und Verwaltung von Quantenalgorithmen und -operationen auf einer abstrakteren Ebene erleichtern.

Mehrere Studien betonen die Vorteile von Open-Source-Ansätzen, die die Zusammenarbeit und den Wissensaustausch in der Forschungsgemeinschaft fördern [ACC+23a; GEKW23]. Open-Source-Ansätze könnten die Entwicklung und Verbreitung von Quantencomputing-Tools beschleunigen.

Zusätzlich wird die Benutzerfreundlichkeit von Low-Code-Plattformen hervorgehoben, um deren Akzeptanz zu erhöhen [PPA+22; PRH21]. Eine benutzerfreundliche Gestaltung könnte entscheidend sein, um die Nutzung von Low-Code-Werkzeuge im Quantencomputing zu fördern.

Diese Trends zeigen, dass die Kombination von Low-Code-Entwicklung und Quantencomputing vielversprechend ist, um die Barrieren für den Einsatz von Quantencomputing zu senken und die Innovationskraft in diesem Bereich zu fördern.

Unterschiede und Besonderheiten in den Ansätzen

Die untersuchten Publikationen zeigen signifikante Unterschiede und Besonderheiten in den Ansätzen zur Integration von Low-Code-Entwicklung in Quantencomputing-Anwendungen, insbesondere unter Berücksichtigung von Model-Driven Engineering (MDE) und Open-Source-Prinzipien.

Ein wesentlicher Unterschied liegt in der Zielsetzung der verschiedenen Low-Code-Plattformen. Während einige Publikationen den Schwerpunkt auf die Vereinfachung und Beschleunigung der Entwicklungsprozesse legen [KKM+20; SSI+20], zielen andere darauf ab, komplexe Quantenalgorithmen und -operationen zugänglicher zu machen [GEKW23; GGG+21]. Diese unterschiedliche Ausrichtung beeinflusst die Auswahl und Gestaltung der Entwicklungswerkzeuge erheblich.

Besonderheiten zeigen sich auch in der Integration von Model-Driven Engineering. Einige Ansätze verwenden MDE, um die Erstellung von Quantenalgorithmen zu automatisieren und zu optimieren [MMC+21; PPP+22]. Andere nutzen MDE, um hybride Systeme zu modellieren, die sowohl klassische als auch Quantenkomponenten enthalten [PTMC24a; WWB+20]. Diese unterschiedlichen Anwendungen von MDE verdeutlichen die Vielfalt und Flexibilität der Methode im Kontext von Quantencomputing.

Ein weiterer Unterschied liegt in der Nutzung von Open-Source-Prinzipien. Während einige Publikationen Open-Source-Ansätze als Mittel zur Förderung von Innovation und Kollaboration hervorheben [AAA23; ACC+23a], betonen andere die Herausforderungen und Einschränkungen, die mit der Entwicklung und Pflege von Open-Source-Tools einhergehen [SSAA21]. Die Entscheidung für oder gegen Open-Source beeinflusst die Zugänglichkeit und Anpassungsfähigkeit der entwickelten Tools maßgeblich.

Diese Unterschiede und Besonderheiten verdeutlichen, dass es keine einheitliche Lösung gibt, sondern dass die Wahl des Ansatzes stark von den spezifischen Anforderungen und Zielen des jeweiligen Projekts abhängt. Es zeigt sich, dass die Kombination von Low-Code-Entwicklung und Quantencomputing durch flexible und anpassbare Ansätze unterstützt werden muss, um den vielfältigen Herausforderungen und Potenzialen gerecht zu werden.

Integration der Erkenntnisse und Bedeutung der Ergebnisse

Die Ergebnisse der systematischen Literaturrecherche tragen zur Beantwortung der Forschungsfrage 3.1 bei, wie Low-Code-Entwicklungsansätze effektiv in die Konzeption und Entwicklung von Quantencomputing-Anwendungen integriert werden können, unter besonderer Berücksichtigung von Model-Driven Engineering (MDE) und Open-Source-Prinzipien.

Die Analyse zeigt, dass Low-Code-Entwicklungsansätze durch die Bereitstellung visueller Entwicklungsumgebungen und die Automatisierung komplizierterer Programmierprozesse einen niedrigschwelligen Zugang zur Entwicklung von Quantenalgorithmen ermöglichen [BBFF21b; SSI+20]. Diese Ansätze können die Einstiegshürden senken und die Technologie für eine breitere Nutzerschicht zugänglich machen, einschließlich derjenigen ohne tiefgehende Programmierkenntnisse.

Model-Driven Engineering (MDE) bietet Möglichkeiten zur Strukturierung und Automatisierung der Entwicklungsprozesse. MDE ermöglicht es, formale Modelle zur Beschreibung von Quantenalgorithmen zu nutzen, welche anschließend in lauffähigen Code transformiert werden können [GGG+21; PPP+22]. Diese methodische Vorgehensweise kann die Effizienz und die Qualität der entwickelten Anwendungen verbessern.

Darüber hinaus betonen die Studien die Bedeutung von Open-Source-Prinzipien. Open-Source-Tools bieten nicht nur die Möglichkeit zur freien Anpassung und Erweiterung, sondern fördern auch die Kollaboration und den Wissenstransfer innerhalb der Entwicklergemeinschaft [AAA23; ACC+23a]. Die Verfügbarkeit von Open-Source-Lösungen könnte zur Entwicklung und zur Verbreitung von Lösungen des Quantencomputings beitragen.

Die Kombination dieser Erkenntnisse legt nahe, dass die Integration von Low-Code-Entwicklungsansätzen und MDE in die Quantencomputing-Entwicklung sowohl technische als auch organisatorische Vorteile bietet. Die Verwendung von Low-Code-Werkzeugen und MDE könnte die Entwicklungszeiten verkürzen, die Qualität der Software erhöhen und eine breitere Akzeptanz und Nutzung von Quantencomputing fördern. Zudem könnte der Einsatz von Open-Source-Software eine flexible und nachhaltige Weiterentwicklung der Software ermöglichen, was langfristig zur Stabilität und zum Wachstum des Quantencomputing-Ökosystems beitragen könnte.

Zusammenfassend zeigen die Ergebnisse, dass ein Low-Code-Framework für Quantencomputing technisch machbar und potenziell vorteilhaft ist, um die Entwicklung und Nutzung von Quantencomputing-Anwendungen zu fördern. Dies bildet eine solide Grundlage für die weitere Forschung und die praktische Umsetzung in zukünftigen Projekten.

Identifikation von Forschungslücken

Die systematische Literaturrecherche hat einige Bereiche identifiziert, in denen bisher nur wenig oder keine Forschung betrieben wurde. Diese Forschungslücken bieten Potenzial für zukünftige wissenschaftliche Untersuchungen und die Weiterentwicklung von Low-Code-Frameworks für Quantencomputing.

Ein Bereich mit deutlichen Forschungslücken ist die konkrete Anwendung von Low-Code-Entwicklungsansätzen im Quantencomputing. Obwohl einige Publikationen die theoretischen Vorteile und möglichen Ansätze diskutieren, fehlt es an empirischen Studien und praktischen Implementierungen, die die tatsächliche Machbarkeit und Effizienz solcher Ansätze evaluieren [GGG+21; PPPP20]. Zukünftige Forschungsarbeiten könnten daher darauf abzielen, konkrete Prototypen zu entwickeln und zu testen, um die praktische Anwendbarkeit von Low-Code-Entwicklungsansätzen im Quantencomputing zu validieren.

Ein weiterer unerforschter Bereich betrifft die Integration von Model-Driven Engineering (MDE) mit Low-Code-Plattformen im Kontext des Quantencomputings. Während MDE für klassische Softwareentwicklung gut etabliert ist, gibt es nur wenige Studien, die sich mit der spezifischen Anwendung von MDE für Quantenalgorithmen und -anwendungen beschäftigen [GEKW23; PPP+22]. Zukünftige Forschung könnte untersuchen, wie MDE-Prinzipien und -Werkzeuge angepasst und optimiert werden können, um die Entwicklung von Quantenanwendungen zu unterstützen.

Auch die Rolle von Open-Source-Software in der Entwicklung von Low-Code-Frameworks für Quantencomputing ist ein Bereich, der weiter erforscht werden sollte. Obwohl die Vorteile von Open-Source-Software, wie die Förderung von Kollaboration und Innovation, anerkannt sind, gibt es nur wenige Studien, die die spezifischen Herausforderungen und Möglichkeiten der Implementierung von Open-Source-Low-Code-Plattformen für Quantencomputing untersuchen [AAA23; ACC+23a]. Zukünftige Arbeiten könnten sich darauf konzentrieren, Strategien zu entwickeln, um Open-Source-Initiativen in diesem Bereich zu fördern und die damit verbundenen technischen und organisatorischen Herausforderungen zu adressieren.

Insgesamt zeigt die Analyse, dass es mehrere vielversprechende Forschungsrichtungen gibt, die weiterverfolgt werden sollten. Dazu gehören die empirische Validierung von Low-Code-Ansätzen im Quantencomputing, die Anpassung und Anwendung von MDE für Quantenalgorithmen sowie die Entwicklung und Förderung von Open-Source-Lösungen in diesem Bereich.

Kritische Analyse der Qualität und Relevanz der gefundenen Studien

Die im Rahmen dieser systematischen Literaturrecherche identifizierten Studien weisen unterschiedliche Qualitäten und Relevanzgrade auf. Eine kritische Analyse der Stärken und Schwächen dieser Studien sowie eine Bewertung der Zuverlässigkeit und Validität der Ergebnisse sind entscheidend für die Ableitung fundierter Erkenntnisse und zukünftiger Forschungsschwerpunkte.

Einige der identifizierten Studien zeichnen sich durch detaillierte theoretische Analysen und umfassende Literaturübersichten aus, die ein tiefes Verständnis der jeweiligen Themenbereiche vermitteln. Beispielsweise bieten [PPPP20] und [GGG+21] fundierte theoretische Grundlagen zu den Prinzipien der Low-Code-Entwicklung und deren Anwendung im Quantencomputing. Diese Studien legen die theoretischen Rahmenbedingungen dar und identifizieren potenzielle Herausforderungen und Chancen.

Jedoch weisen einige Studien auch Schwächen auf. Ein häufiges Problem ist die begrenzte empirische Validierung der vorgestellten Konzepte und Modelle. Viele Arbeiten konzentrieren sich auf theoretische Betrachtungen und bieten nur wenige oder keine praktischen Implementierungen oder Fallstudien, um die vorgeschlagenen Ansätze zu verifizieren [ACC+23a]. Dies führt zu einer eingeschränkten Aussagekraft hinsichtlich der tatsächlichen Anwendbarkeit und Effektivität der diskutierten Methoden.

Zuverlässigkeit und Validität der Ergebnisse

Die Zuverlässigkeit und Validität der Ergebnisse variieren ebenfalls stark zwischen den Studien. Einige Arbeiten, wie [AAA23], verwenden robuste methodische Ansätze und bieten detaillierte Beschreibungen ihrer Forschungsdesigns, was die Nachvollziehbarkeit und Reproduzierbarkeit ihrer Ergebnisse erhöht. Diese Studien liefern wertvolle Einsichten und können als zuverlässige Informationsquellen betrachtet werden.

Auf der anderen Seite gibt es Studien, die methodische Mängel aufweisen oder nur begrenzte Datenquellen nutzen. Solche Arbeiten können aufgrund von unzureichender Datenbasis oder methodischen Schwächen zu fragwürdigen oder wenig generalisierbaren Ergebnissen führen [KKM+20]. Es ist wichtig, diese Limitationen zu erkennen und bei der Interpretation der Ergebnisse zu berücksichtigen.

Insgesamt ist festzustellen, dass die Qualität und Relevanz der identifizierten Studien variieren. Während einige Arbeiten wertvolle theoretische und methodische Beiträge leisten, besteht ein klarer Bedarf an weiteren empirischen Untersuchungen und praktischen Implementierungen, um die Zuverlässigkeit und Anwendbarkeit der vorgestellten Konzepte und Modelle zu bestätigen. Zukünftige Forschungsarbeiten sollten daher verstärkt empirische Validierungen einschließen und methodische Strenge anstreben, um die Qualität und Aussagekraft der Ergebnisse zu erhöhen.

Schlussfolgerungen basierend auf der Synthese

Die Synthese der identifizierten Studien liefert wertvolle Erkenntnisse für die Konzeption und Entwicklung eines Low-Code-Frameworks für Quantencomputing-Anwendungen. Die wichtigsten Schlussfolgerungen und daraus abgeleiteten Empfehlungen sind wie folgt:

Die Integration von Model-Driven Engineering (MDE) kann die Entwicklung von Quantencomputing-Anwendungen erheblich erleichtern. Durch die Verwendung formaler Modelle und automatisierter Transformationen kann der Entwicklungsprozess effizienter gestaltet und die Qualität der resultierenden Anwendungen verbessert werden [FR07a; Sel03]. Ein Modellierungswerkzeug, das es ermöglicht, Quantenprogramme auf hoher Abstraktionsebene zu entwerfen und automatisch in ausführbaren Code zu transformieren, ist hierbei besonders vorteilhaft.

Ein zentrales Ziel des Low-Code-Ansatzes ist es, die Barrieren für die Nutzung von Quantencomputing zu senken und diese einem breiteren Spektrum von Nutzern zugänglich zu machen. Dies wird durch die Bereitstellung intuitiver, grafischer Entwicklungswerkzeuge erreicht, die auch von Nicht-Experten genutzt werden können. Die Ergebnisse zeigen, dass visuelle Programmierung und benutzerfreundliche Schnittstellen entscheidend sind, um die Akzeptanz und Verbreitung von Low-Code-Werkzeuge im Quantencomputing zu fördern [KKM+20].

Die Untersuchung hat gezeigt, dass Open-Source-Lösungen wichtige Vorteile bieten, darunter erhöhte Zugänglichkeit, Anpassungsfähigkeit und Innovationskraft. Daher wird empfohlen, das entwickelte Framework als Open-Source-Tool bereitzustellen, um eine breite Nutzung und kontinuierliche Weiterentwicklung durch die Community zu ermöglichen. Open-Source könnte auch dazu beitragen, mehr Transparenz zu schaffen und die Zusammenarbeit zwischen Forschung und Industrie zu fördern [CCC20].

Bei der Entwicklung des Low-Code-Frameworks für Quantencomputing sollten folgende spezifische Punkte berücksichtigt werden:

- **Visuelle Programmierung:** Eine benutzerfreundliche grafische Oberfläche, die es ermöglicht, Quantenprogramme durch Drag-and-Drop-Mechanismen und visuelle Elemente zu erstellen, ist essentiell.
- **Abstraktionsebenen:** Verschiedene Abstraktionsebenen sollten bereitgestellt werden, um sowohl Einsteigern als auch erfahrenen Entwicklern gerecht zu werden. Dies umfasst einfache visuelle Blöcke für Anfänger und detaillierte Konfigurationsmöglichkeiten für Experten.
- **Fehlerkorrektur:** Mechanismen zur Fehlerkorrektur und Optimierung der Quantenoperationen sollten integriert werden, um die Robustheit und Leistungsfähigkeit der Anwendungen zu gewährleisten.
- **Modell-zu-Code-Transformation:** Eine zuverlässige Modell-zu-Code-Transformation, die es ermöglicht, von Modellen direkt ausführbaren Quanten-Code zu generieren, ist von großer Bedeutung.
- **Plattformintegration:** Die nahtlose Integration in bestehende Quantencomputing-Plattformen sollte gewährleistet sein, um die Kompatibilität und Nutzung vorhandener Ressourcen zu optimieren.
- **Skalierbarkeit:** Die Skalierbarkeit des Frameworks ist zu berücksichtigen, um auch bei komplexen Quantenanwendungen eine hohe Performance zu gewährleisten.
- **Benutzerfreundlichkeit:** Umfassende Dokumentationen und Tutorials sollten entwickelt werden, um die Einarbeitung und Nutzung des Frameworks zu erleichtern.
- **Sicherheitsaspekte:** Sicherheitsaspekte sind bei der Entwicklung zu berücksichtigen, insbesondere im Kontext der Quantenkryptographie und der sicheren Datenübertragung.
- **Community-Feedback:** Kontinuierliches Feedback aus der Community sollte integriert werden, um das Framework iterativ zu verbessern und an die Bedürfnisse der Nutzer anzupassen.
- **Open-Source-Bereitstellung:** Das Framework sollte als Open-Source-Software bereitgestellt werden, um die Zusammenarbeit und Innovation innerhalb der Community zu fördern.

Durch die Berücksichtigung dieser Aspekte kann ein wertvolles Werkzeug geschaffen werden, das die Entwicklung von Quantenanwendungen erleichtert und die breitere Nutzung dieser fördert. Diese Empfehlungen basieren auf der Synthese der identifizierten Studien und sollen als Leitlinien für die Entwicklung eines effektiven und benutzerfreundlichen Low-Code-Frameworks für Quantencomputing-Anwendungen dienen.

4 Zusammenfassung und Ausblick

Diese Arbeit befasste sich mit der Konzeption eines Rahmens für ein Low-Code-Framework für Quantencomputing-Anwendungen. Die Motivation für diese Untersuchung liegt in der inhärenten Komplexität der Quantenprogrammierung, die eine hohe Einstiegshürde für Entwickler darstellt, und der vielversprechenden Möglichkeit, diese Barrieren durch Low-Code-Ansätze zu senken.

Zu Beginn wurde ein umfassender Überblick über die theoretischen Grundlagen der Low-Code-Entwicklung, des Model-Driven Engineerings und des Quantencomputings gegeben. Dabei wurde erläutert, wie Low-Code-Entwicklungsumgebungen durch die Abstraktion technischer Komplexitäten und die Bereitstellung intuitiver, grafischer Entwicklungswerkzeuge die Zugänglichkeit zur Softwareentwicklung erhöhen können. Ebenso wurden die Potenziale und Herausforderungen von MDE in der Softwareentwicklung beschrieben.

Die systematische Literaturrecherche, die im Rahmen dieser Arbeit durchgeführt wurde, identifizierte zahlreiche Studien, die relevante Ansätze und Tools im Bereich Low-Code und Quantencomputing beleuchteten. Dabei wurden sowohl Gemeinsamkeiten als auch Unterschiede in den verschiedenen Ansätzen herausgearbeitet. Es zeigte sich, dass viele der bestehenden Low-Code-Entwicklungsplattformen bereits einige Aspekte der Quantenprogrammierung unterstützen, jedoch noch signifikante Forschungslücken bestehen. Im Rahmen der Arbeit wurde aufgezeigt wie ein prototypisches Low-Code-Framework für Quantencomputing konzipiert werden kann, das auf den Erkenntnissen der Literaturrecherche basiert.

Die Analyse der identifizierten Publikationen zeigte mehrere Forschungslücken auf. Insbesondere besteht ein Bedarf an der Entwicklung robuster Low-Code-Tools, die eine breite Anwendbarkeit und Anpassungsfähigkeit bieten. Diese Anforderung gilt gleichermaßen für Open-Source-Lösungen wie auch für kommerzielle Produkte. Weiterhin ist es erforderlich, die Effizienz und Skalierbarkeit solcher Low-Code-Frameworks im Kontext des Quantencomputings zu verbessern.

Insgesamt zeigt diese Arbeit, dass die Integration von Low-Code-Entwicklungsansätzen in die Quantencomputing-Entwicklung zwar vielversprechend ist, jedoch noch in einem frühen Stadium steckt. Zukünftige Forschung sollte sich darauf konzentrieren, die identifizierten Lücken zu schließen und bestehende Frameworks gezielt zu optimieren. Ein konkreter Ansatzpunkt für die weitere Forschung wäre die prototypische Implementierung eines solchen Low-Code-Frameworks für Quantencomputing, das in bestehende Entwicklungsumgebungen und Pipelines integriert wird. Dabei sollten insbesondere die Transformation und Ausführung von Quantenalgorithmen berücksichtigt werden, um die Nutzungsmöglichkeiten für eine breitere Entwicklergemeinschaft zu erweitern.

Literaturverzeichnis

[AAA23] A. Ahmad, A. B. Altamimi, J. M. Aqib. „A Reference Architecture for Quantum Computing as a Service“. In: *Journal of King Saud University: Computer and Information Sciences* (2023). DOI: 10.48550/arxiv.2306.04578 (zitiert auf S. 43–46, 62).

[AAM+21] A. A. Alamin, A. A. Alamin, S. Malakar, S. Malakar, G. Uddin, G. Uddin, S. Afroz, S. Afroz, S. Afroz, T. B. Haider, T. B. Haider, A. Iqbal, A. Iqbal. „An Empirical Study of Developer Discussions on Low-Code Software Development Challenges“. In: *null* (2021). DOI: 10.1109/msr52588.2021.00018 (zitiert auf S. 19–21).

[AAS+22] D. Alonso, D. Alonso, P. Sánchez, P. Sánchez, F. Sánchez-Rubio, F. Sánchez-Rubio. „Engineering the development of quantum programs: Application to the Boolean satisfiability problem“. In: *Advances in Engineering Software* (2022). DOI: 10.1016/j.advengsoft.2022.103216 (zitiert auf S. 63).

[AAU+22] M. A. A. Alamin, M. A. A. Alamin, G. Uddin, G. Uddin, S. Malakar, S. Malakar, S. Afroz, S. Afroz, T. Haider, T. B. Haider, A. Iqbal, A. Iqbal. „Developer discussion topics on the adoption and barriers of low code software development platforms“. In: *Empirical Software Engineering* (2022). DOI: 10.1007/s10664-022-10244-0 (zitiert auf S. 61).

[AAYY20] S. Ali, S. Ali, T. Yue, T. Yue. „Modeling Quantum programs: challenges, initial results, and research directions“. In: *APEQESESEC/SIGSOFT FSE* (2020). DOI: 10.1145/3412451.3428499 (zitiert auf S. 63).

[ACC+23a] F. Amato, M. Cicalese, L. Contrasto, G. Cubicciotti, G. D'Ambola, A. L. Marca, G. Pagano, F. Tomeo, G. A. Robertazzi, G. Vassallo, G. Acampora, A. Vitiello, G. Catolino, G. Giordano, S. Lambiase, V. Pontillo, G. Sellitto, F. Ferrucci, F. Palomba. „Quantumoonlight: A Low-Code Platform to Experiment with Quantum Machine Learning“. In: *SoftwareX* (2023). DOI: 10.2139/ssrn.4333822 (zitiert auf S. 38–40, 42–45).

[ACC+23b] F. Amato, M. Cicalese, L. Contrasto, G. Cubicciotti, G. D'Ambola, A. La Marca, G. Pagano, F. Tomeo, G. A. Robertazzi, G. Vassallo et al. „QuantuMoonLight: A low-code platform to experiment with quantum machine learning“. In: *SoftwareX* 22 (2023), S. 101399 (zitiert auf S. 62).

[ACS+24] I. Alfonso, A. Conrardy, A. Sulejmani, A. Nirumand, F. Ul Haq, M. Gomez-Vazquez, J.-S. Sottet, J. Cabot. „Building BESSER: an open-source low-code platform“. In: *International Conference on Business Process Modeling, Development and Support*. Springer. 2024, S. 203–212 (zitiert auf S. 32, 33, 61).

[ADLH05] A. Aspuru-Guzik, A. D. Dutoi, P. J. Love, M. Head-Gordon. „Simulated quantum computation of molecular energies“. In: *Science* 309.5741 (2005), S. 1704–1707 (zitiert auf S. 19).

[ADR82] A. Aspect, J. Dalibard, G. Roger. „Experimental test of Bell's inequalities using time-varying analyzers“. In: *Physical review letters* 49.25 (1982), S. 1804 (zitiert auf S. 18).

[ARK22] M. A. Akbar, S. Rafi, A. A. Khan. „Classical to Quantum Software Migration Journey Begins: A Conceptual Readiness Model“. In: *null* (2022). DOI: 10.1007/978-3-031-21388-5_42 (zitiert auf S. 63).

[ASÁ23] D. Alonso, P. Sánchez, B. Álvarez. „A Graph-Based Approach for Modelling Quantum Circuits“. In: *Applied Sciences* 13.21 (2023), S. 11794 (zitiert auf S. 63).

[BB14] C. H. Bennett, G. Brassard. „Quantum cryptography: Public key distribution and coin tossing“. In: *Theoretical computer science* 560 (2014), S. 7–11 (zitiert auf S. 19).

[BBFF21a] A. C. Bock, A. Böck, U. Frank, U. Frank. „In Search of the Essence of Low-Code: An Exploratory Study of Seven Development Platforms“. In: *2021 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)* (2021). DOI: 10.1109/models-c53483.2021.00016 (zitiert auf S. 38–40, 42, 61).

[BBFF21b] A. Böck, A. Bock, U. Frank, U. Frank. „Low-Code Platform“. In: *Business & Information Systems Engineering* (2021). DOI: 10.1007/s12599-021-00726-8 (zitiert auf S. 19, 21, 42, 44).

[BCW17] M. Brambilla, J. Cabot, M. Wimmer. *Model-driven software engineering in practice*. Morgan & Claypool Publishers, 2017 (zitiert auf S. 20, 23).

[BD23] P. Bocciarelli, A. D'Ambrogio. „A Low-Code Approach for Simulation-Based Analysis of Process Collaborations“. In: *Online World Conference on Soft Computing in Industrial Applications* (2023). DOI: 10.1109/wsc60868.2023.10407452 (zitiert auf S. 61).

[BKB+07] P. Brereton, B. A. Kitchenham, D. Budgen, M. Turner, M. Khalil. „Lessons from applying the systematic literature review process within the software engineering domain“. In: *Journal of systems and software* 80.4 (2007), S. 571–583 (zitiert auf S. 29, 30).

[BSW22] A. Büscher, D. Schilberg, L. Wiegert. „The Use of Low-Code During a Skill Shortage“. In: *ASME International Mechanical Engineering Congress and Exposition*. Bd. 86649. American Society of Mechanical Engineers. 2022, V02BT02A027 (zitiert auf S. 13).

[CAB+21] M. Cerezo, A. Arrasmith, R. Babbush, S. Benjamin, S. Endo, K. Fujii, J. McClean, K. Mitarai, X. Yuan, L. Cincio, P. Coles. „Variational quantum algorithms“. In: *Nature Reviews Physics* 3.9 (Sep. 2021), S. 625–644 (zitiert auf S. 13).

[CB23] V. Cole, M. Boutet. „ResearchRabbit“. In: *The Journal of the Canadian Health Libraries Association* 44.2 (2023), S. 43 (zitiert auf S. 31).

[CCC20] J. Cabot, J. Cabot, J. Cabot. „Positioning of the low-code movement within the field of model-driven engineering“. In: *ACM/IEEE International Conference on Model Driven Engineering Languages and Systems* (2020). DOI: 10.1145/3417990.3420210 (zitiert auf S. 14, 22, 26, 46).

[CFK+22] S. Chitransh, D. Fischer, N. Kawalek, N. LaRacuente, J. Markman, S. Gaunkar, U. Zvi. „A Multidisciplinary, Artistic Approach to Broadening the Accessibility of Quantum Science“. In: *2022 IEEE International Conference on Quantum Computing and Engineering (QCE)*. IEEE. 2022, S. 701–708 (zitiert auf S. 13).

[Che06] H. Chesbrough. *Open business models: How to thrive in the new innovation landscape*. Harvard Business Press, 2006 (zitiert auf S. 26).

[DJ92] D. Deutsch, R. Jozsa. „Rapid solution of problems by quantum computation“. In: *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences* 439.1907 (1992), S. 553–558 (zitiert auf S. 19).

[DKL+22] D. Di Ruscio, D. Kolovos, J. de Lara, A. Pierantonio, M. Tisi, M. Wimmer. „Low-code development and model-driven engineering: Two sides of the same coin?“ In: *Software and Systems Modeling* 21.2 (2022), S. 437–446 (zitiert auf S. 22, 26).

[EPR35] A. Einstein, B. Podolsky, N. Rosen. „Can quantum-mechanical description of physical reality be considered complete?“ In: *Physical review* 47.10 (1935), S. 777 (zitiert auf S. 17).

[Fey18] R. P. Feynman. „Simulating physics with computers“. In: *Feynman and computation*. cRc Press, 2018, S. 133–153 (zitiert auf S. 17, 18).

[FGGS00] E. Farhi, J. Goldstone, S. Gutmann, M. Sipser. „Quantum computation by adiabatic evolution“. In: *arXiv preprint quant-ph/0001106* (2000) (zitiert auf S. 19).

[Fit06] B. Fitzgerald. „The transformation of open source software“. In: *MIS quarterly* (2006), S. 587–598 (zitiert auf S. 25).

[FR07a] R. France, B. Rumpe. „Model-driven Development of Complex Software: A Research Roadmap“. In: *Future of Software Engineering (FOSE '07)*. 2007, S. 37–54. DOI: 10.1109/FOSE.2007.14 (zitiert auf S. 21, 23, 46).

[FR07b] R. France, B. Rumpe. „Model-driven development of complex software: A research roadmap“. In: *Future of Software Engineering (FOSE'07)*. IEEE. 2007, S. 37–54 (zitiert auf S. 22).

[GEKW23] F. Gemeinhardt, M. Eisenberg, S. Klikovits, M. Wimmer. „Model-Driven Optimization for Quantum Program Synthesis with MOMoT“. In: *2023 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)* (2023). DOI: 10.1109/models-c59198.2023.00100 (zitiert auf S. 42, 43, 45, 63).

[GGG+21] F. Gemeinhardt, F. Gemeinhardt, A. Garmendía, A. Garmendia, M. Wimmer, M. Wimmer. „Towards Model-Driven Quantum Software Engineering“. In: *Workshop on Quantum Software Engineering* (2021). DOI: 10.1109/q-se52541.2021.00010 (zitiert auf S. 42–45).

[GGW21a] F. Gemeinhardt, A. Garmendia, M. Wimmer. „Towards Model-Driven Quantum Software Engineering“. In: *2021 IEEE/ACM 2nd International Workshop on Quantum Software Engineering (Q-SE)*. 2021, S. 13–15. DOI: 10.1109/Q-SE52541.2021.00010 (zitiert auf S. 13, 14, 62).

[GGW21b] F. Gemeinhardt, A. Garmendia, M. Wimmer. „Towards model-driven quantum software engineering“. In: *2021 IEEE/ACM 2nd International Workshop on Quantum Software Engineering (Q-SE)*. IEEE. 2021, S. 13–15 (zitiert auf S. 40, 42).

[GGWW18] F. Gemeinhardt, A. Garmendía, M. Wimmer, R. Wille. „A Model-Driven Framework for Composition-Based Quantum Circuit Design“. In: *null* (2018). DOI: null (zitiert auf S. 40, 62).

[Gro96] L. K. Grover. „A fast quantum mechanical algorithm for database search“. In: *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*. 1996, S. 212–219 (zitiert auf S. 19).

[GS18] D. J. Griffiths, D. F. Schroeter. *Introduction to quantum mechanics*. Cambridge university press, 2018 (zitiert auf S. 18).

[HH21] G. Hurlburt, G. Hurlburt. „Low-Code, No-Code, What's Under the Hood?“ In: *IT Professional* (2021). DOI: 10.1109/mitp.2021.3123415 (zitiert auf S. 61).

[JJP+22] L. Jiménez-Navajas, L. Jimenez-Navajas, R. Pérez-Castillo, R. Perez-Castillo, M. Piattini, M. Piattini. „Transforming Quantum Programs in Kdm to Quantum Design Models in Uml“. In: *Social Science Research Network* (2022). DOI: 10.2139/ssrn.4074848 (zitiert auf S. 63).

[JMJ+22] G. Juhás, L. Molnár, A. Juhásová, M. Ondrišová, M. Mladoniczky, T. Kováčik. „Low-code platforms and languages: the future of software development“. In: *2022 20th International Conference on Emerging eLearning Technologies and Applications (ICETA)*. IEEE. 2022, S. 286–293 (zitiert auf S. 13).

[JPP21] L. Jiménez-Navajas, R. Pérez-Castillo, M. Piattini. „Kdm to uml model transformation for quantum software modernization“. In: *Quality of Information and Communications Technology* (2021). DOI: 10.1007/978-3-030-85347-1_16 (zitiert auf S. 63).

[JPP23] L. Jiménez-Navajas, R. Pérez-Castillo, M. Piattini. „Reverse Engineering of OpenQASM3 Quantum Programs to KDM Models“. In: *International Conference on Evaluation of Novel Approaches to Software Engineering* (2023). DOI: 10.5220/0011963000003464 (zitiert auf S. 63).

[KAW+23] A. A. Khan, A. Ahmad, M. Waseem, P. Liang, M. Fahmideh, T. Mikkonen, P. Abrahamsson. „Software architecture for quantum computing systems—A systematic review“. In: *Journal of Systems and Software* 201 (2023), S. 111682 (zitiert auf S. 63).

[KC+07] B. Kitchenham, S. Charters et al. *Guidelines for performing systematic literature reviews in software engineering*. 2007 (zitiert auf S. 29).

[KKA+22] A. A. Khan, A. A. Khan, A. Ahmad, A. Ahmad, M. Waseem, M. Waseem, P. Liang, P. Liang, M. Fahmideh, M. Fahmideh, T. Mikkonen, T. Mikkonen, P. Abrahamsson, P. Abrahamsson. „Software Architecture for Quantum Computing Systems - A Systematic Review“. In: *Social Science Research Network* (2022). DOI: 10.2139/ssrn.4191449 (zitiert auf S. 62).

[KKB+19] N. Kahani, N. Kahani, M. Bagherzadeh, M. Bagherzadeh, J. R. Cordy, J. R. Cordy, J. Dingel, J. Dingel, D. Varró, D. Varró. „Survey and classification of model transformation tools“. In: *Software and Systems Modeling* (2019). DOI: 10.1007/s10270-018-0665-6 (zitiert auf S. 62).

[KKM+20] F. Khorram, F. Khorram, J.-M. Mottu, J.-M. Mottu, G. Sunyé, G. Sunyé. „Challenges & opportunities in low-code testing“. In: *null* (2020). DOI: 10.1145/3417990.3420204 (zitiert auf S. 39, 42, 43, 46, 61).

[KSW22] S. Kass, S. Strahringer, M. Westner. „Drivers and Inhibitors of Low Code Development Platform Adoption“. In: *null* (2022). DOI: 10.1109/cbi54897.2022.00028 (zitiert auf S. 61).

[Let21] T. C. Lethbridge. „Low-code is often high-code, so we must design low-code platforms to enable proper software engineering“. In: *Leveraging Applications of Formal Methods, Verification and Validation: 10th International Symposium on Leveraging Applications of Formal Methods, ISoLA 2021, Rhodes, Greece, October 17–29, 2021, Proceedings 10*. Springer. 2021, S. 202–212 (zitiert auf S. 61).

[LJL+10] T. D. Ladd, F. Jelezko, R. Laflamme, Y. Nakamura, C. Monroe, J. L. O'Brien. „Quantum computers“. In: *nature* 464.7285 (2010), S. 45–53 (zitiert auf S. 19).

[MBC23] A. Moin, A. Badii, M. Challenger. „Model-Driven Quantum Federated Learning (QFL)“. In: *null* (2023). DOI: 10.1145/3594671.3594690 (zitiert auf S. 39, 40, 62).

[MGM+24] J. M. Murillo, J. Garcia-Alonso, E. Moguel, J. Barzen, F. Leymann, S. Ali, T. Yue, P. Arcaini, R. Pérez, I. G. R. de Guzmán et al. „Challenges of quantum software engineering for the next decade: The road ahead“. In: *arXiv preprint arXiv:2404.06825* (2024) (zitiert auf S. 62).

[Min22] N. Minerbi. „Quantum Software Development with Classiq“. In: *Quantum Software Engineering*. Springer, 2022, S. 269–280 (zitiert auf S. 32, 62).

[MK13] C. Monroe, J. Kim. „Scaling the ion trap quantum processor“. In: *Science* 339.6124 (2013), S. 1164–1169 (zitiert auf S. 27).

[MMC+21] A. Moin, A. Moin, M. Challenger, M. Challenger, A. Badii, A. Badii, S. Günnemann, S. Günnemann. „MDE4QAI: Towards Model-Driven Engineering for Quantum Artificial Intelligence“. In: *arXiv: Software Engineering* (2021). DOI: null (zitiert auf S. 42, 43, 63, 64).

[MR22] M. Motta, J. Rice. „Emerging quantum computing algorithms for quantum chemistry“. In: *Wiley Interdisciplinary Reviews: Computational Molecular Science* 12.3 (2022), e1580 (zitiert auf S. 14).

[NC10] M. A. Nielsen, I. L. Chuang. *Quantum computation and quantum information*. Cambridge university press, 2010 (zitiert auf S. 17, 18, 27).

[OS15] C. Okoli, K. Schabram. „A guide to conducting a systematic literature review of information systems research“. In: (2015) (zitiert auf S. 29).

[PFMM08] K. Petersen, R. Feldt, S. Mujtaba, M. Mattsson. „Systematic mapping studies in software engineering“. In: *12th international conference on evaluation and assessment in software engineering (EASE)*. BCS Learning & Development. 2008 (zitiert auf S. 29).

[PJCP23] R. Pérez-Castillo, L. Jiménez-Navajas, I. Cantalejo, M. Piattini. „Generation of Classical-Quantum Code from UML models“. In: *International Conference on Quantum Computing and Engineering* (2023). DOI: 10.1109/qce57702.2023.10202 (zitiert auf S. 63).

[PP22] R. Pérez-Castillo, M. Piattini. „Design of classical-quantum systems with UML“. In: *Computing* 104.11 (2022), S. 2375–2403 (zitiert auf S. 62).

[PPA+22] D. Pinho, D. Pinho, A. Aguiar, A. Aguiar, V. Amaral, V. Amaral. „What about the usability in low-code platforms? A systematic literature review“. In: *Journal of computer languages* (2022). DOI: 10.1016/j.cola.2022.101185 (zitiert auf S. 38, 40, 43, 61).

[PPP+21a] R. Pérez-Castillo, R. Pérez-Castillo, R. Pérez-Castillo, L. Jiménez-Navajas, L. Jiménez-Navajas, M. Piattini, M. Piattini, M. Piattini. „Modelling Quantum Circuits with UML“. In: *null* (2021). DOI: 10.1109/q-se52541.2021.00009 (zitiert auf S. 64).

[PPP+21b] M. Piattini, M. Piattini, M. Piattini, M. A. Serrano, M. Serrano, R. Pérez-Castillo, R. Pérez-Castillo, R. Pérez-Castillo, G. Petersen, G. Petersen, J. L. Hevia, J. L. Hevia. „Toward a Quantum Software Engineering“. In: *IT Professional* (2021). DOI: 10.1109/mitp.2020.3019522 (zitiert auf S. 63).

[PPP+22] R. Pérez-Castillo, R. Pérez-Castillo, R. Pérez-Castillo, M. Piattini, M. Piattini. „Design of classical-quantum systems with UML“. In: *Computing* (2022). DOI: 10.1007/s00607-022-01091-4 (zitiert auf S. 42–45).

[PPPP20] C. A. Pérez-Delgado, C. A. Pérez-Delgado, H. G. Pérez-González, H. G. Perez-Gonzalez. „Towards a Quantum Software Modeling Language“. In: *null* (2020). DOI: 10.1145/3387940.3392183 (zitiert auf S. 40, 44, 45, 62).

[Pre18] J. Preskill. „Quantum computing in the NISQ era and beyond“. In: *Quantum* 2 (2018), S. 79 (zitiert auf S. 18).

[PRH21] N. Prinz, C. Rentrop, M. Huber. „Low-Code Development Platforms - A Literature Review“. In: *Americas Conference on Information Systems* (2021). DOI: null (zitiert auf S. 39, 43, 61).

[PTMC24a] F. Polat, H. Tuncer, A. Moin, M. Challenger. „Model-Driven Engineering for Quantum Programming: A Case Study on Ground State Energy Calculation“. In: *arXiv.org* (2024). DOI: 10.48550/arxiv.2405.17065 (zitiert auf S. 43).

[PTMC24b] F. Polat, H. Tuncer, A. Moin, M. Challenger. „Model-Driven Engineering for Quantum Programming: A Case Study on Ground State Energy Calculation“. In: *arXiv preprint arXiv:2405.17065* (2024) (zitiert auf S. 63).

[Ray10] E. S. Raymond. *The cathedral and the bazaar*. 2010 (zitiert auf S. 25).

[RP11] E. Rieffel, W. Polak. *Quantum computing: A gentle introduction*. MIT Press, März 2011 (zitiert auf S. 13).

[RTK+24] N. Rao, J. Tsay, K. Kate, V. Hellendoorn, M. Hirzel. „AI for Low-Code for AI“. In: *Proceedings of the 29th International Conference on Intelligent User Interfaces*. IUI '24. Greenville, SC, USA, 2024, S. 837–852. ISBN: 9798400705083 (zitiert auf S. 38, 40, 61).

[SBMP08] D. Steinberg, F. Budinsky, E. Merks, M. Paternostro. *EMF: eclipse modeling framework*. Pearson Education, 2008 (zitiert auf S. 21, 23).

[Sch06] D. Schmidt. „Guest Editor's Introduction: Model-Driven Engineering“. In: *Computer* 39.2 (2006), S. 25–31. DOI: 10.1109/MC.2006.58 (zitiert auf S. 20, 23).

[Sel03] B. Selic. „The pragmatics of model-driven development“. In: *IEEE Software* 20.5 (2003), S. 19–25. DOI: 10.1109/MS.2003.1231146 (zitiert auf S. 20, 23, 46).

[Sho99a] P. Shor. „Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer“. In: *SIAM Review* 41.2 (1999), S. 303–332 (zitiert auf S. 13).

[Sho99b] P. W. Shor. „Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer“. In: *SIAM review* 41.2 (1999), S. 303–332 (zitiert auf S. 18, 19).

[SS21] S. Shridhar, S. Shridhar. „Analysis of Low Code-No Code Development Platforms in comparison with Traditional Development Methodologies“. In: *International Journal for Research in Applied Science and Engineering Technology* (2021). DOI: 10.22214/ijraset.2021.39328 (zitiert auf S. 19–21).

[SSAA21] P. Sánchez, P. Sánchez, D. Alonso, D. Alonso. „On the Definition of Quantum Programming Modules“. In: *Applied Sciences* (2021). DOI: 10.3390/app11135843 (zitiert auf S. 43, 62).

[SSI+20] A. Sahay, A. Sahay, A. Indamutsa, A. Indamutsa, D. D. Ruscio, D. D. Ruscio, A. Pierantonio, A. Pierantonio. „Supporting the understanding and comparison of low-code development platforms“. In: *EUROMICRO Conference on Software Engineering and Advanced Applications* (2020). DOI: 10.1109/seaa51224.2020.00036 (zitiert auf S. 38, 39, 42–44, 61).

[SSN+22] M. D. Stefano, M. D. Stefano, D. D. Nucci, D. D. Nucci, F. Palomba, F. Palomba, D. Taibi, D. Taibi, A. D. Lucia, A. D. Lucia. „Towards Quantum-algorithms-as-a-service“. In: *null* (2022). DOI: 10.1145/3549036.3562056 (zitiert auf S. 62).

[VV06] G. Von Krogh, E. Von Hippel. „The promise of research on open source software“. In: *Management science* 52.7 (2006), S. 975–983 (zitiert auf S. 26).

[Weba] E. Web. *Eclipse Acceleo — projects.eclipse.org*. [Accessed 12-08-2024]. URL: https://projects.eclipse.org/projects/modeling.acceleo (zitiert auf S. 21).

[Webb] E. Web. *Eclipse EMF — projects.eclipse.org*. [Accessed 12-08-2024]. URL: https://projects.eclipse.org/projects/modeling.emf.emf (zitiert auf S. 21).

[WG06] J. West, S. Gallagher. „Challenges of open innovation: the paradox of firm investment in open-source software“. In: *R&d Management* 36.3 (2006), S. 319–331 (zitiert auf S. 26).

[WWB+20] B. Weder, B. Weder, U. Breitenbücher, U. Breitenbücher, F. Leymann, F. Leymann, K. Wild, K. Wild. „Integrating Quantum Computing into Workflow Modeling and Execution“. In: *null* (2020). DOI: 10.1109/ucc48980.2020.00046 (zitiert auf S. 43, 62).

Alle URLs wurden zuletzt am 18. 07. 2024 geprüft.

Anhang

Anhang

Anhang 1: Übersicht der Publikationen

Tabelle A1: Übersicht der Publikationen

ID	Titel	Autoren	Jahr	Themengebiet
P01	In Search of the Essence of Low-Code: An Exploratory Study of Seven Development Platforms	Alexander C. Bock, Ulrich Frank [BBFF21a]	2021	Low-Code
P02	What about the usability in low-code platforms? A systematic literature review	Daniel Pinho et al. [PPA+22]	2022	Low-Code
P03	Low-Code Development Platforms - A Literature Review	Niculin Prinz et al. [PRH21]	2021	Low-Code
P04	Supporting the understanding and comparison of low-code development platforms	Apurvanand Sahay et al. [SSI+20]	2020	Low-Code
P05	Building BESSER: an open-source low-code platform	Iván Alfonso et al. [ACS+24]	2024	Low-Code
P06	AI for Low-Code for AI	Nikitha Rao et al. [RTK+24]	2024	Low-Code
P07	Challenges & opportunities in low-code testing	Faezeh Khorram et al. [KKM+20]	2020	Low-Code
P08	Low-Code Is Often High-Code, So We Must Design Low-Code Platforms to Enable Proper Software Engineering	Timothy C. Lethbridge [Let21]	2021	Low-Code
P09	Low-Code, No-Code, What's Under the Hood?	George Hurlburt [HH21]	2021	Low-Code
P10	Drivers and Inhibitors of Low Code Development Platform Adoption	Sebastian Kass et al. [KSW22]	2022	Low-Code
P11	A Low-Code Approach for Simulation-Based Analysis of Process Collaborations	Paolo Bocciarelli, Andrea D'Ambrogio [BD23]	2023	Low-Code, Model-Driven Engineering
P12	Developer discussion topics on the adoption and barriers of low code software development platforms	Md Abdullah Al Alamin et al. [AAU+22]	2022	Low-Code, Model-Driven Engineering

Tabelle A1: Übersicht der Publikationen

ID	Titel	Autoren	Jahr	Themengebiet
P13	Quantumoonlight: A Low-Code Platform to Experiment with Quantum Machine Learning	Francesco Amato et al. [ACC+23b]	2023	Low-Code, Quantum Computing
P14	Towards a Quantum Software Modeling Language	Carlos A. Pérez-Delgado et al. [PPPP20]	2020	Low-Code, Quantum Computing, Model-Driven Engineering
P15	Model-Driven Quantum Federated Learning (QFL)	Armin Moin et al. [MBC23]	2023	Low-Code, Quantum Computing, Model-Driven Engineering
P16	Towards Model-Driven Quantum Software Engineering	Felix Gemeinhardt et al. [GGW21a]	2021	Low-Code, Quantum Computing, Model-Driven Engineering
P17	A Model-Driven Framework for Composition-Based Quantum Circuit Design	Felix Gemeinhardt et al. [GGWW18]	2018	Low-Code, Quantum Computing, Model-Driven Engineering
P18	A Reference Architecture for Quantum Computing as a Service	Aakash Ahmad et al. [AAA23]	2023	Low-Code, Quantum Computing, Model-Driven Engineering
P19	Design of classical-quantum systems with UML	Ricardo Pérez-Castillo et al. [PP22]	2022	Low-Code, Quantum Computing, Model-Driven Engineering
P20	Integrating Quantum Computing into Workflow Modeling and Execution	Benjamin Weder et al. [WWB+20]	2020	Low-Code, Quantum Computing, Model-Driven Engineering
P21	Towards Quantum-algorithms-as-a-service	Manuel De Stefano et al. [SSN+22]	2022	Low-Code, Quantum Computing, Model-Driven Engineering
P22	Survey and classification of model transformation tools	Nafıseh Kahani et al. [KKB+19]	2019	Model-Driven Engineering
P23	Quantum Software Development with Classiq	Nir Minerbi [Min22]	2022	Quantum Computing
P24	Challenges of Quantum Software Engineering for the Next Decade: The Road Ahead	J. M. Murillo et al. [MGM+24]	2024	Quantum Computing
P25	Software Architecture for Quantum Computing Systems - A Systematic Review	Arif Ali Khan et al. [KKA+22]	2022	Quantum Computing
P26	On the Definition of Quantum Programming Modules	Pedro Sánchez, Diego Alonso [SSAA21]	2021	Quantum Computing

Tabelle A1: Übersicht der Publikationen

ID	Titel	Autoren	Jahr	Themengebiet
P27	Generation of Classical-Quantum Code from UML models	Ricardo Pérez-Castillo et al. [PJCP23]	2023	Quantum Computing, Model-Driven Engineering
P28	Model-Driven Optimization for Quantum Program Synthesis with MOMoT	Felix Gemeinhardt et al. [GEKW23]	2023	Quantum Computing, Model-Driven Engineering
P29	Reverse Engineering of OpenQASM3 Quantum Programs to KDM Models	Luis Jiménez-Navajas et al. [JPP23]	2023	Quantum Computing, Model-Driven Engineering
P30	A Graph-Based Approach for Modelling Quantum Circuits	Diego Alonso et al. [ASÁ23]	2023	Quantum Computing, Model-Driven Engineering
P31	MDE4QAI: Towards Model-Driven Engineering for Quantum Artificial Intelligence	Moin, Armin et al. [MMC+21]	2021	Quantum Computing, Model-Driven Engineering
P32	Engineering the development of quantum programs: Application to the Boolean satisfiability problem	Diego Alonso et al. [AAS+22]	2022	Quantum Computing, Model-Driven Engineering
P33	Model-Driven Engineering for Quantum Programming: A Case Study on Ground State Energy Calculation	Furkan Polat et al. [PTMC24b]	2024	Quantum Computing, Model-Driven Engineering
P34	Kdm to uml model transformation for quantum software modernization	Luis Jiménez-Navajas et al. [JPP21]	2021	Quantum Computing, Model-Driven Engineering
P35	Modeling Quantum programs: challenges, initial results, and research directions	Shaukat Ali et al. [AAYY20]	2020	Quantum Computing, Model-Driven Engineering
P36	Transforming Quantum Programs in Kdm to Quantum Design Models in Uml	Luis Jiménez-Navajas et al. [JJP+22]	2022	Quantum Computing, Model-Driven Engineering
P37	Toward a Quantum Software Engineering	Mario Piattini et al. [PPP+21b]	2021	Quantum Computing, Model-Driven Engineering
P38	Software architecture for quantum computing systems - A systematic review	Arif Ali Khan et al. [KAW+23]	2023	Quantum Computing, Model-Driven Engineering
P39	Classical to Quantum Software Migration Journey Begins: A Conceptual Readiness Model	Muhammad Azeem Akbar et al. [ARK22]	2022	Quantum Computing, Model-Driven Engineering

Tabelle A1: Übersicht der Publikationen

ID	Titel	Autoren	Jahr	Themengebiet
P40	Modelling Quantum Circuits with UML	Ricardo Pérez-Castillo et al. [PPP+21a]	2021	Quantum Computing, Model-Driven Engineering
P41	MDE4QAI: Towards Model-Driven Engineering for Quantum Artificial Intelligence	Armin Moin et al. [MMC+21]	2021	Quantum Computing, Model-Driven Engineering

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift